Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Identificação de snoRNAs utilizando Máquinas de Vetores de Suporte: aprimorando o snoReport

João Victor de Araujo Oliveira

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Orientadora
Prof.ª Dr.ª Maria Emília M. Telles Walter

Coorientador
Prof. Dr. Pedro de Azevedo Berger

Brasília
2014

Universidade de Brasília — UnB
Instituto de Ciências Exatas
Departamento de Ciência da Computação
Bacharelado em Ciência da Computação

Coordenadora: Prof.ª Dr.ª Maristela Terto Holanda

Banca examinadora composta por:

Prof.ª Dr.ª Maria Emília M. Telles Walter (Orientadora) — CIC/UnB
Prof. Dr. Pedro de Azevedo Berger — CIC/UnB
Prof.ª Dr.ª Tainá Raiol Alencar — IB/UnB

CIP — Catalogação Internacional na Publicação

Oliveira, João Victor de Araujo.

Identificação de snoRNAs utilizando Máquinas de Vetores de Suporte: aprimorando o snoReport / João Victor de Araujo Oliveira. Brasília : UnB, 2014.
163 p. : il. ; 29,5 cm.

Monografia (Graduação) — Universidade de Brasília, Brasília, 2014.

1. Bioinformática, 2. Aprendizagem de Máquina, 3. SVM, 4. ncRNA, 5. snoRNA

CDU 004.4

Endereço: Universidade de Brasília
Campus Universitário Darcy Ribeiro — Asa Norte
CEP 70910-900
Brasília–DF — Brasil

Universidade de Brasília
Instituto de Ciências Exatas
Departamento de Ciência da Computação

Identificação de snoRNAs utilizando Máquinas de Vetores de Suporte: aprimorando o snoReport

João Victor de Araujo Oliveira

Monografia apresentada como requisito parcial
para conclusão do Bacharelado em Ciência da Computação

Prof.ª Dr.ª Maria Emília M. Telles Walter (Orientadora)
CIC/UnB

Prof. Dr. Pedro de Azevedo Berger
CIC/UnB

Prof.ª Dr.ª Tainá Raiol Alencar
IB/UnB

Prof.ª Dr.ª Maristela Terto Holanda
Coordenadora do Bacharelado em Ciência da Computação

Brasília, 20 de Fevereiro de 2014

Dedicatória

Dedico este trabalho, primeiramente a Deus, pois sem ele eu nada seria, por ele nunca ter me abandonado nestes 5 anos de muito estudo na UnB.

Também dedico este trabalho a todos os meus familiares, por tudo que fizeram por mim, me incentivando sempre a estudar e me dando todo apoio que eu precisava. Minha namorada, por todo carinho, atenção e paciência nesses últimos meses de TCC.

Aos meus amigos, principalmente meus colegas de faculdade e do laboratório de bioinformática. Sofremos juntos, mas vencemos juntos!

Aos meus orientadores: Maria Emília e Pedro Berger, por toda força e ensinamentos que me deram neste período de TCC. Muito obrigado!

"Faça o necessário, depois o possível, e, de repente, você estará fazendo o impossível." São Francisco de Assis.

Agradecimentos

Além das pessoas que já citei na dedicatória, quero agradecer a todas aquelas pessoas que, com gestos simples, me ajudaram em toda a minha jornada acadêmica. Muito obrigado!

Resumo

O estudo de RNAs não-codificadores (*ncRNAs*) vem se mostrando bastante desafiador devido a dificuldade de verificar experimentalmente qual função desempenha determinado gene não codificador de proteína em um organismo [20]. Métodos computacionais para a identificação de *ncRNAs* possuem dificuldades similares aos métodos experimentais, pois a Bioinformática não possui um método único para a identificação e classificação de ncRNAS. Em particular, métodos que utilizam Aprendizagem de Máquina tem se mostrado bastante úteis para essa tarefa. Um método computacional que vem mostrando bons resultados é o snoReport [30], que combina predição de estrutura secundária de *snoRNAs* com Máquina de Vetores de Suporte (SVM) para identificação de *snoRNAs*, uma classe específica de *ncRNA* que contém duas classes principais, *snoRNA* H/ACA box e *snoRNA* C/D box. Entretanto, o snoReport necessita de um refinamento pois está disponível hoje um volume bem maior de snoRNAs, alguns já testados experimentalmente. Neste contexto, este trabalho tem como objetivo aprimorar a identificação de *snoRNAs* usando o banco de dados disponibilizados pelo Centro de Bioinformática da Universidade de Leipzig, além das versões mais novas de programas usados no *workflow* do snoReport. Essas alterações levaram a índices de 30,43% de melhoria na sensibilidade do classificador de *snoRNAs* H/ACA box e de 24,24% de melhoria na sensibilidade do classificador *snoRNA* C/D box. Além disso, foi feito um estudo de caso com o fungo *Paracoccidioides brasiliensis*, que identificou 188 *snoRNAs* C/D box e 5.929 *snoRNAs* H/ACA box

Palavras-chave: Bioinformática, Aprendizagem de Máquina, SVM, ncRNA, snoRNA

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Os estudos de Watson e Crick [73] permitiram estabelecer a estrutura espacial da molécula de DNA. Essa descoberta serviu como base para os atuais projetos de sequenciamento genético, que vem possibilitando ampliar os conhecimentos das funções e mecanismos genéticos, fisiológicos e metabólicos dos mais diversos organismos [20]. A Biologia Molecular é a área responsável pelo estudo de ácidos nucléicos, estrutura de proteínas, processos relacionados e outros atores envolvidos, tais como organelas celulares e enzimas [68].Os ácidos nucléicos têm a função principal de armazenar informação necessária e prover mecanismos para a produção de proteínas e também de possibilitar a transferência desta informação para outros organismos, através de processos de reprodução celular [68]. Na natureza encontramos dois tipos de ácidos nucléicos, o DNA (ácido desoxirribonucléico) e o RNA (ácido ribonucléico). Proteínas são moléculas formadas por um conjunto de aminoácidos e exercem uma vasta quantidade de funções vitais para os seres vivos, tais como acelerar reações químicas (como é o caso das enzimas), transporte de nutrientes, eliminação de resíduos tóxicos e também construir estruturas complexas [5].

O avanço dos estudos de Watson e Crick permitiu que fosse proposto o Dogma Central da Biologia Molecular [73], que mostra como determinadas regiões da molécula de DNA, através do processo de transcrição, são transformadas em uma molécula de RNA mensageiro (*mRNA*) e este, pelo processo de tradução, é sintetizado em uma proteína, por meio de dois RNAs, o ribossomal (*rRNA*) e o transportador (*tRNA*).

Contudo, grande parte do material genético não codifica proteínas, por exemplo, no ser humano, apenas 2% do seu genoma gera RNAs codificadores de proteína [69]. Tais regiões são denominadas RNAs não-codificadores de proteínas (*ncRNAs*), que hoje são objeto de pesquisa de vários cientistas de todo o mundo. Por meio de experimentos biológicos, foi possível perceber uma relação de ncRNAs com a regulação gênica e com funções estruturais, sendo algumas delas relacionadas com a transcrição e tradução de *mRNAs*, como o *tRNA* e o *rRNA* [20].

Alguns *ncRNAs* recém-gerados na transcrição sofrem um processamento posterior, gerando RNAs menores e metabolicamente estáveis possuindo diversas funções, como os *rRNAs, tRNAs* e *snRNAs*[1]. Os *ncRNAs* gerados por esse pós-processamento são então modificados por um outro *ncRNA* denominado *snoRNA* [24, 51], objeto de estudo deste trabalho. os *snoRNAs*, em células de eucariotos, participam de um processo denominado

[1] *snRNAs* são envolvidos no processo de *splicing*.

splicing, em que porções do *mRNA* (introns) são removidas, e as porções restantes (exons) são então ligadas formando o *mRNA* maduro.

Para dar suporte aos projetos de sequenciamento da Biologia Molecular, devido a grande quantidade de dados a serem produzidos e analisados, surgiu a Bioinformática, uma área que, de acordo com o *Nacional Institute of Health* (NIH), pode ser definida como a "pesquisa, desenvolvimento ou aplicação de ferramentas computacionais e abordagens para expandir o uso de dados biológicos, médicos, comportamentais ou de saúde, incluindo adquirir, armazenar, organizar, analisar ou visualizar tais dados" [60].

Devido a grande quantidade de problemas computacionais estudados na Bioinformática, uma grande variedade de métodos computacionais de diferentes áreas da Ciência da Computação foram criados para resolver tais desafios. Particularmente uma subárea da Inteligência Artificial a ser explorada para resolver problemas em Bioinformática é a Aprendizagem de Máquina (*Machine Learning*). A área de Aprendizagem de Máquina tem como foco a questão de como automaticamente construir programas de computadores que são aperfeiçoados com a experiência [54].

Há quatro tipos de aprendizagem de máquina: supervisionada, não-supervisionada, semi-supervisionada e aprendizagem por reforço.

Na aprendizagem supervisionada, há um conjunto de treinamento onde cada amostra possui uma característica ou classe já conhecida e com ela é então gerada uma hipótese que será usada para descobrir as classes em que uma nova amostra pertence.

Já na aprendizagem não-supervisionada, há um conjunto de treinamento, contudo suas amostras não estão ainda classificadas ou caracterizadas completamente, tendo este aprendizado como objetivo de encontrar alguma estrutura, ou padrão, escondido no conjunto de treinamento.

No caso da aprendizagem semi-supervisionada, nem todas as amostras possuem uma classe conhecida, o que é útil em casos onde não se possui uma grande quantidade de amostras já classificadas anteriormente.

Por fim na aprendizagem por reforço, não há um conjunto de treinamento e o sistema deve se adaptar a partir de um julgamento de suas ações que podem ser positivas ou negativas e, a partir daí, tomar decisões que favoreçam determinado objetivo.

Um algoritmo bastante utilizado na aprendizagem supervisionada é a Máquina de Vetores de Suporte (*SVM*). Uma *SVM* é uma máquina linear que tem como principal ideia, no contexto de problemas de classificação de padrões, construir um hiperplano como superfície de decisão de tal modo que a margem de separação entre amostras positivas e negativas é maximizada [29]. Isto é feito na fase de treinamento, em que duas ou mais classes serão separadas a partir de uma função. A fase de testes, permite avaliar a sensibilidade e especificidade de dados ainda não classificados, mas que têm suas classes conhecidas.

1.1 Motivação

O estudo de *ncRNA* vem se mostrando bastante desafiador devido a dificuldade de verificar experimentalmente qual função desempenha determinado gene não codificador em um organismo [20]. Métodos computacionais possuem dificuldades similares aos métodos experimentais, sendo que a Bioinformática não possui um método único para a identificação e classificação de ncRNAs. Este problema se dá primordialmente devido

a essas moléculas possuírem uma grande associação entre sua estrutura tridimensional e sua função, inviabilizando métodos utilizados anteriormente para detecção de *mRNAs* codificadores de proteínas, baseado apenas nas suas sequências.

Um método computacional que vem mostrando bons resultados é o snoReport [30], que combina predição de estrutura secundária com SVM para identificar *snoRNAs*. O snoReport é capaz de reconhecer *snoRNAs* em sequências individuais, sem incluir informação sobre o RNA alvo complementar ao bloco antisenso, o que inviabilizaria a descoberta de alguns *snoRNAs* órfãos que não possuem complementaridade com o RNA alvo [35, 36].

1.2 Problema

O método utilizado pelo snoReport [30] precisa de refinamento, pois existem novas versões de todos as ferramentas utilizadas e está disponível hoje um volume bem maior de snoRNAs, alguns experimentalmente verificados.

1.3 Objetivos

1.3.1 Objetivo principal

O objetivo geral deste trabalho é aprimorar a fase de treinamento da SVM usada no snoReport, usando as novas versões das ferramentas e os dados atualizados de *ncRNAs* disponibilizados pelo Centro de Bioinformática da Universidade de Leipzig.

1.3.2 Objetivos específicos

- Executar o *workflow* do snoReport, utilizando as versões mais novas do pacote Vienna RNA (v2.16) [32] e da libSVM (v3.17) [13];
- Treinar a SVM do snoReport com dados disponibilizados pelo Centro de Bioinformática da Universidade de Leipzig;
- Realizar um estudo de caso com o fungo *Paracoccidioides brasiliensis* para identificar *snoRNAs* ainda não conhecidos.

1.4 Descrição dos capítulos

O capítulo 2 apresenta conceitos básicos de Biologia Molecular e Bioinformática, com foco em *ncRNAs*. Em seguida, são exploradas as funções e classificações dos *ncRNAs* nos organismos, além de métodos e banco de dados para identificação de *ncRNAs*. Em particular, descrevemos características de *snoRNAs* e apresentamos as ferramentas snoReport e snoSeeker.

No capítulo 3, são apresentados os principais conceitos e abordagens utilizadas na Aprendizagem de Máquina, particularmente SVM.

No capítulo 4, é apresentado o método proposto para a identificação de *snoRNAs*, aprimorando a fase de treinamento do snoReport tanto para *snoRNAs* H/ACA box quanto para *snoRNAs* C/D box.

No capítulo 5, são discutidos os resultados do snoReport reimplementado, além de serem mostrados os novos *snoRNAs* identificados no fungo *Paracoccidioides brasiliensis*.

Por fim, no capítulo 6, o trabalho é concluído e são apresentadas sugestões para trabalhos futuros.

Capítulo 2

RNAs não-codificadores

Neste capítulo serão apresentados conceitos básicos de Biologia Molecular e Bioinformática, particularmente detalhes de RNAs não codificadores (*ncRNAs*). Na seção 2.1 serão descritos: conceitos sobre o DNA, RNA e proteínas, além do Dogma Central da Biologia Molecular. Na seção 2.2 serão apresentados: a definição de *ncRNAs*; a classificação de diferentes famílias de *ncRNAs*; a definição de snoRNAs; bancos de dados de *ncRNAs* e por fim métodos para a predição de *ncRNAs*, em particular o snoReport [30].

2.1 Biologia Molecular e Bioinformática

A Biologia Molecular é o ramo da Biologia responsável pelo estudo de ácidos nucléicos, estrutura de proteínas, processos relacionados e outros atores envolvidos, tais como organelas celulares e enzimas [68]. Para dar suporte aos estudos referentes à Biologia Molecular, devido a enorme quantidade de dados produzidos por sequenciadores automáticos a serem analisados, surgiu a Bioinformática, uma nova área que, de acordo com o *National Institute of Health* (NIH), pode ser definida como a "pesquisa, desenvolvimento ou aplicação de ferramentas computacionais e abordagens para expandir o uso de dados biológicos, médicos, comportamentais ou de saúde, incluindo adquirir, armazenar, organizar, analisar ou visualizar tais dados" [60].

2.1.1 DNA e RNA

Os ácidos nucléicos têm a função principal de armazenar informação necessária e prover mecanismos para a criação de proteínas e também de possibilitar a transferência dessas informações para outros organismos, através de processos de reprodução celular [68].

Na natureza encontramos dois tipos de ácidos nucléicos, o DNA (ácido desoxirribonucléico) e o RNA (ácido ribonucléico). Essas macromoléculas são formadas por monômeros chamados nucleotídeos. Um nucleotídeo é formado por um açúcar composto por 5 átomos de carbono (pentose), que se liga a um grupo fosfato e a uma base nitrogenada. As diferenças encontradas entre a molécula de DNA e de RNA no nível estrutural são: o RNA possui como pentose a ribose e o DNA possui a desoxirribose. As pentoses do DNA e do RNA podem se ligar a quatro diferentes tipos de bases nitrogenadas que são: A - Adenina, C - Citosina, G - Guanina e T - Timina, caso seja DNA, ou U - Uracila, caso se trate de um RNA, no lugar da Timina.

O DNA é formado por uma dupla fita, disposta espacialmente em formato helicoidal, enquanto o RNA é formado apenas por uma fita. A figura 2.1 mostra a diferença entre a pentose encontrada no DNA (desoxirribose) e a pentose ligada ao RNA (ribose) que consiste na presença ou ausência de uma hidroxila (OH) no carbono 2'.

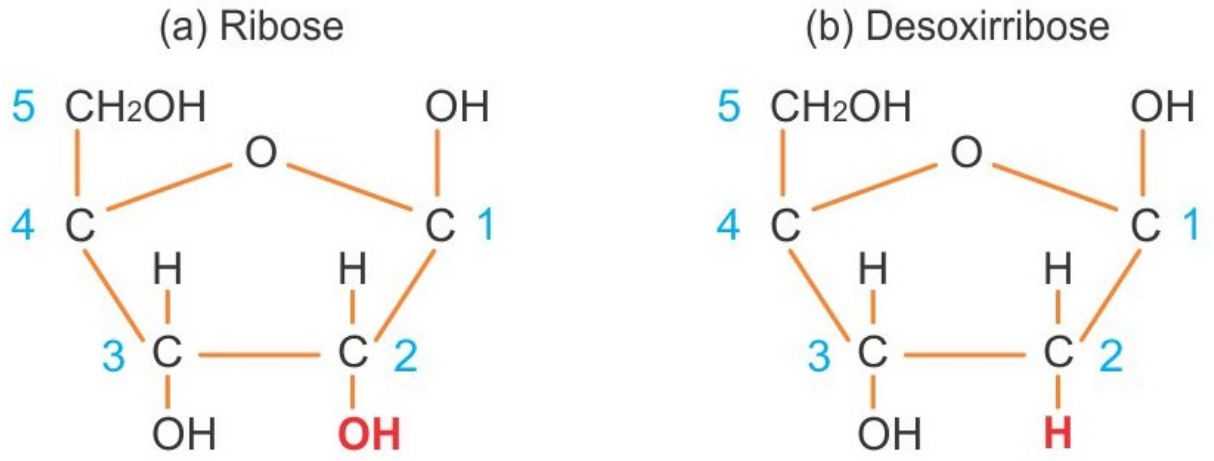


Figura 2.1: Diferença entre as pentoses presentes no RNA e no DNA, respectivamente. (a) A ribose possui uma hidroxila (OH) ligada ao carbono 2'. (b) A desoxirribose possui um átomo de hidrogênio (H) ligado ao carbono 2' [5].

A ligação entre diferentes nucleotídeos para a formação de um ácido nucléico é feita através dos grupos fosfatos, por meio de uma ligação chamada fosfodiéster [5]. Esta ligação possibilita a ligação do carbono 3' de um nucleotídeo a um grupo fosfato, que se liga ao carbono 5' de outro nucleotídeo. É por esta razão que convencionalmente os ácidos nucléicos são sintetizados na direção canônica 5' $\rightarrow$ 3'.

Como dito anteriormente, moléculas de DNA são formadas por duas fitas complementares na forma helicoidal, onde uma fita com sentido 5' $\rightarrow$ 3' (fita codificadora) que liga a uma fita no sentido 3' $\rightarrow$ 5' (fita molde) como se pode ver na figura 2.2. Essas fitas são unidas através de ligação de pontes de hidrogênio devido a complementaridade par a par entre duas bases nitrogenadas. A base A é dita complementar à T (e vice versa) e a base C complementar à G (e vice versa), isto é, esses pares de bases complementares ligam-se através de pontes de hidrogênio, fazendo assim a ligação entre as duas fitas de DNA. Estes pares são conhecidos como *Watson-Crick base pairs* [68] ou simplesmente pares de bases. Pares de bases provêm uma unidade de medida amplamente utilizada para descrever o tamanho de uma molécula de DNA ou RNA e pode ser abreviada para *pb* [68].

Quanto ao nível funcional destas moléculas, o DNA possui como função principal o armazenamento de informações necessárias para a síntese de proteínas ou de *ncRNAs* em um organismo. Essas informações que codificam transcritos estão contidas em regiões do DNA denominadas genes. Existem transcritos codificadores de proteína e aqueles não codificadores de proteína, chamados de ncRNAs. Já o RNA possui diversas funções em um organismo tais como a constituição do ribossomo (rRNA), o transporte de aminoácido (tRNA), carregar as informações para a síntese de proteína (mRNA), e diversos papéis em processos de regulação gênica [5].

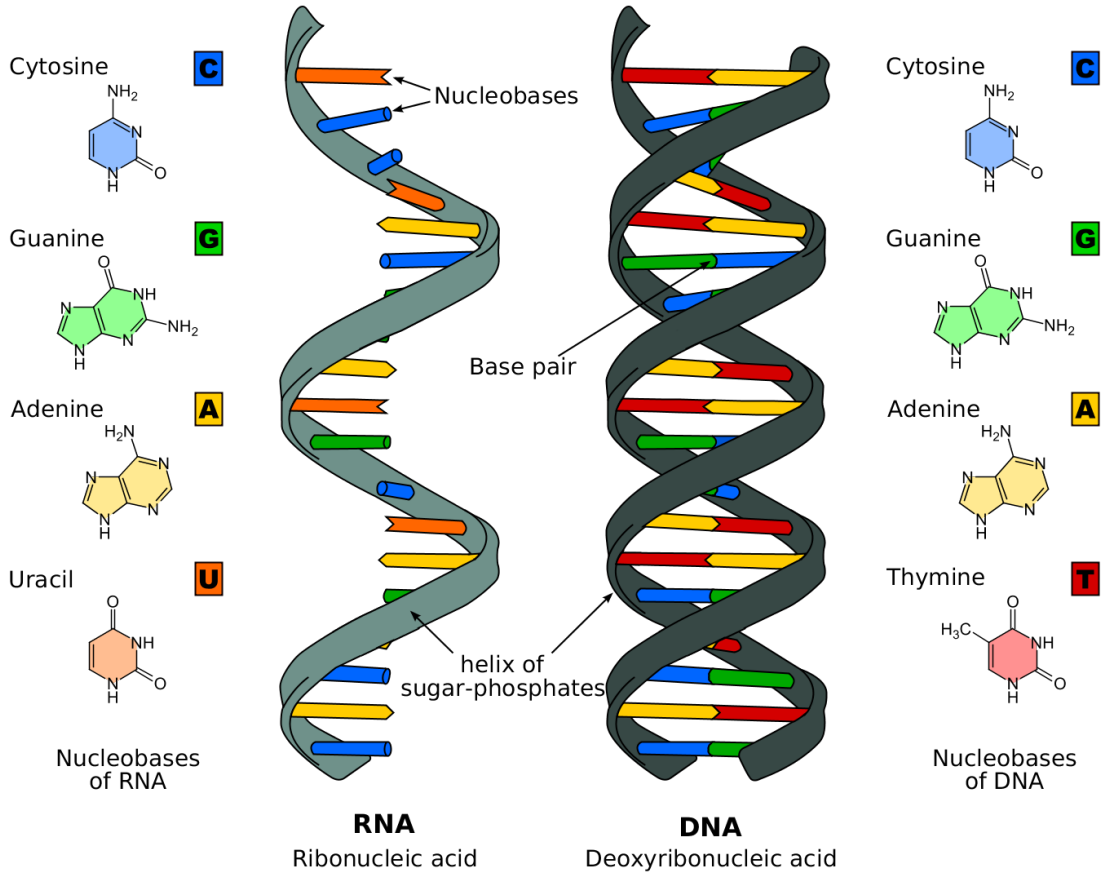


Figura 2.2: Estrutura espacial das Moléculas de RNA e DNA [74].

2.1.2 Dogma Central da Biologia Molecular

O Dogma Central da Biologia Molecular (figura 2.3) refere-se a três processos: replicação, onde uma molécula de DNA é duplicada; transcrição, onde uma porção da fita de DNA traz informações que permitem gerar uma fita de RNA; e tradução, onde o RNA gerado na transcrição será utilizado como molde para a síntese de uma proteína.

A replicação inicia-se com a separação das fitas de DNA. A RNA polimerase identificará na fita molde (3'→5') uma região codificadora de proteína, denominada gene. A RNA polimerase reconhece essa região, que é normalmente precedida por uma sequência característica, denominada promotora, que geralmente possui uma sequência de TA (chamada de *TATA box*) [14]. Tendo identificado a região promotora, a RNA polimerase irá conduzir a transcrição do DNA em um RNA mensageiro (*mRNA*) ou um RNA transcrito. A transcrição ocorre no sentido 5'→3' onde os nucleotídeos A, T, C, G são traduzidos para T, U, G, C, respectivamente.

Neste ponto, vale ressaltar que, caso o organismo seja procarioto (como é o caso das bactérias), o RNA mensageiro produzido já estará pronto para ser utilizado na tradução. Caso seja um organismo eucarioto (como é o caso dos seres humanos, por exemplo), haverá um processo denominado *splicing*. Basicamente no processo de *splicing* irá ocorrer a remoção de regiões gênicas, chamadas de introns, e a ligação das regiões codificadoras denominadas exons.Os introns removidos por esse processo são removidos em forma

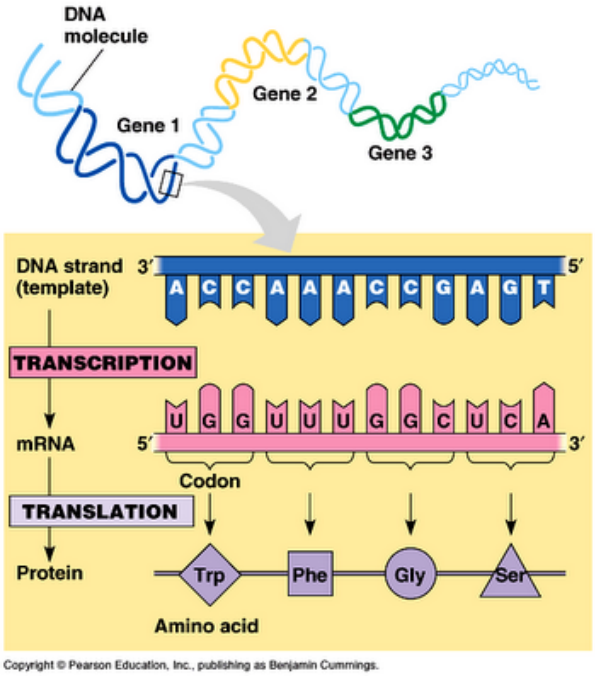


Figura 2.3: Dogma Central da Biologia Molecular [17].

de laços que são abertos e subsequentemente degradados, porém *ncRNAs* denominados *snoRNAs*, que atuam na modificação de *rRNAs, tRNAs* e *snRNAs*, escapam desta degradação formando um complexo protéico [24]. Existe também o *splicing alternativo*, uma combinação entre exons não contíguos, gerando uma diversidade de possíveis transcritos a partir de um mesmo gene.

Após o processo de *splicing*, seja ele alternativo ou não, podem ser inseridos ou deletados alguns nucleotídeos, fenômeno conhecido como *RNA editing* [14]. Este procedimento pode ocorrer pois a sequência de nucleotídeos a ser produzida deve corresponder com determinado padrão a fim de ser produzida uma proteína posteriormente.

Terminada a transcrição, é iniciado o processo de tradução, onde o *mRNA* é sintetizado em uma proteína. Neste ponto, é necessário relembrar que uma proteína é formada basicamente por aminoácidos, e tais aminoácidos serão transportados ao rRNA pelo RNA transportador (*tRNA*), como pode ser visto na figura 2.4. Em uma extremidade de um *tRNA* há um aminoácido e na outra extremidade há uma sequência de três nucleotídeos, denominado anticódon. Este anticódon será ligado a outra trinca denominada códon existente no *mRNA*. A correspondência entre um aminoácido presente em um *tRNA* e um códon é chamado de código genético, apresentado na figura 2.5.

A síntese do *mRNA* ligado a *tRNAs* ocorre nos ribossomos, que são complexos citoplasmáticos constituídos de RNAs ribossomais (*rRNAs*) e proteínas. Os ribossomos funcionam como uma linha de montagem de uma fábrica usando como entradas o mRNA e o tRNA e como saída uma cadeia linear de uma proteína.

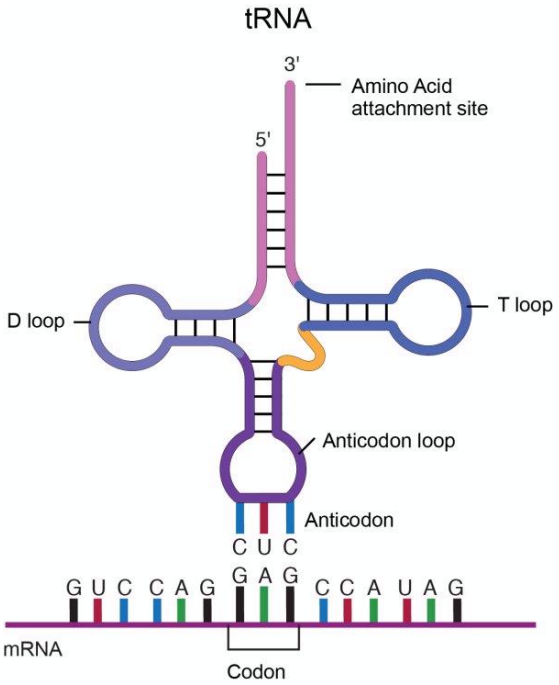


Figura 2.4: Estrutura molecular de um RNA transportador [44].

CÓDONS - m RNA

Primeira Letra	Segunda Letra U	Segunda Letra C	Segunda Letra A	Segunda Letra G	Terceira Letra
U	UUU Phe UUC Phe UUA Leu UUG Leu	UCU Ser UCC Ser UCA Ser UCG Ser	UAU Tyr UAC Tyr UAA Sem sentido UAG Sem sentido	UGU Cys UGC Cys UGA Sem sentido UGG Try	U C A G
C	CUU Leu CUC Leu CUA Leu CUG Leu	CCU Pro CCC Pro CCA Pro CCG Pro	CAU His CAC His CAA Gln CAG Gln	CGU Arg CGC Arg CGA Arg CGG Arg	U C A G
A	AUU Ileu AUC Ileu AUA Ileu AUG Met	ACU Thr ACC Thr ACA Thr ACG Thr	AAU Asn AAC Asn AAA Lys AAG Lys	AGU Ser AGC Ser AGA Arg AGG Arg	U C A G
G	GUU Val GUC Val GUA Val GUG Val	GCU Ala GCC Ala GCA Ala GCG Ala	GAU Asp GAC Asp GAA Glu GAG Glu	GGU Gly GGC Gly GGA Gly GGG Gly	U C A G

Figura 2.5: Código genético mapeando códons para aminoácidos [1].

2.2 RNAs não-codificadores

Projetos genomas iniciados no século XX, tal como o Projeto Genoma Humano [43, 70] tinham normalmente como objetivo identificar sistematicamente genes [23]. Contudo os métodos utilizados para essa identificação, não eram capazes de identificar todas as classes de genes, em particular genes não-codificadores de proteínas, ou *non-coding RNAs* (*ncRNA*) *genes*. Os *ncRNAs* tem funções estruturais, catalíticas ou regulatórias, e não produzem *mRNAs*, que geram proteínas.

Por muito tempo, regiões não codificadoras de proteína eram denominadas de *DNA lixo*, por não haver uma razão particular para estarem lá, a não ser para proteger os genes. Contudo, várias pesquisas vem mostrando que estas regiões desempenham papéis importantíssimos nos organismos [6, 20, 24, 30, 68]. Sabe-se hoje que existe uma relação intrínseca entre a quantidade de material não codificador no DNA e a complexidade de um organismo, como pode ser evidenciado pela comparação da porcentagem de sequências que codificam proteínas entre diferentes organismos. Por exemplo, bactérias, organismos eucariotos unicelulares, invertebrados e mamíferos possuem respectivamente em média 95%, 30%, 20% e 2% de material codificador de proteínas [20].

RNAs não-codificadores que atuam em atividades regulatórias têm sido identificados em todos os domínios de vida e estão envolvidos em numerosos mecanismos de controles de expressão gênica em todos os níveis de transmissão da informação genética do DNA para a criação de uma proteína [69]. Com isso, o Dogma Central da Biologia Molecular, que considerava o RNA apenas com papel intermediário na produção de proteínas, teve de ser revisto incluindo novas tarefas executadas pelos ncRNAs. A figura 2.6 descreve algumas atividades desenvolvidas por *ncRNAs* no Dogma Central da Biologia Molecular.

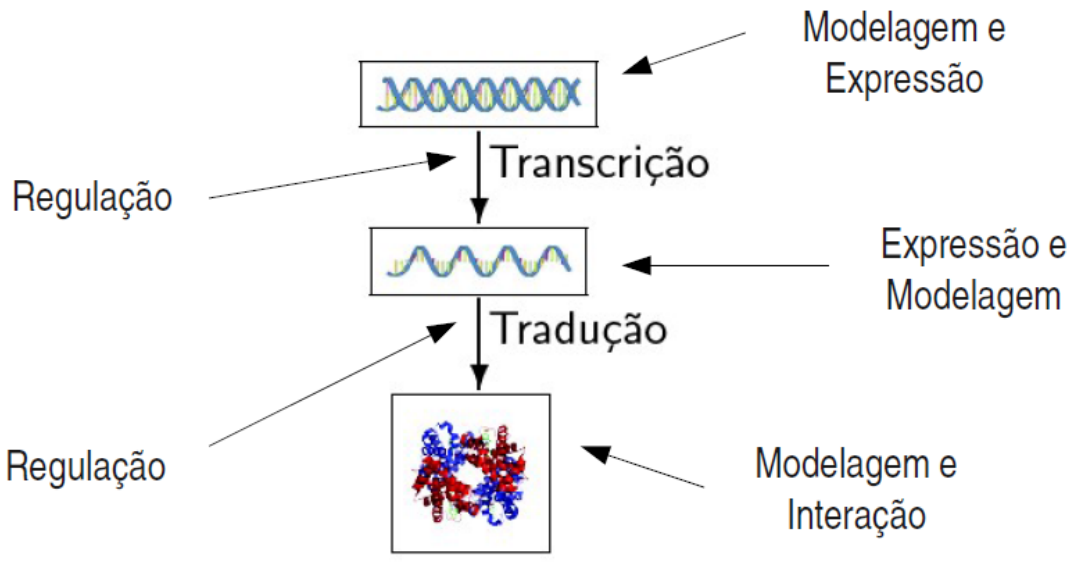


Figura 2.6: Dogma Central da Biologia Molecular acrescida de algumas funções exercidas por *ncRNAs* [20].

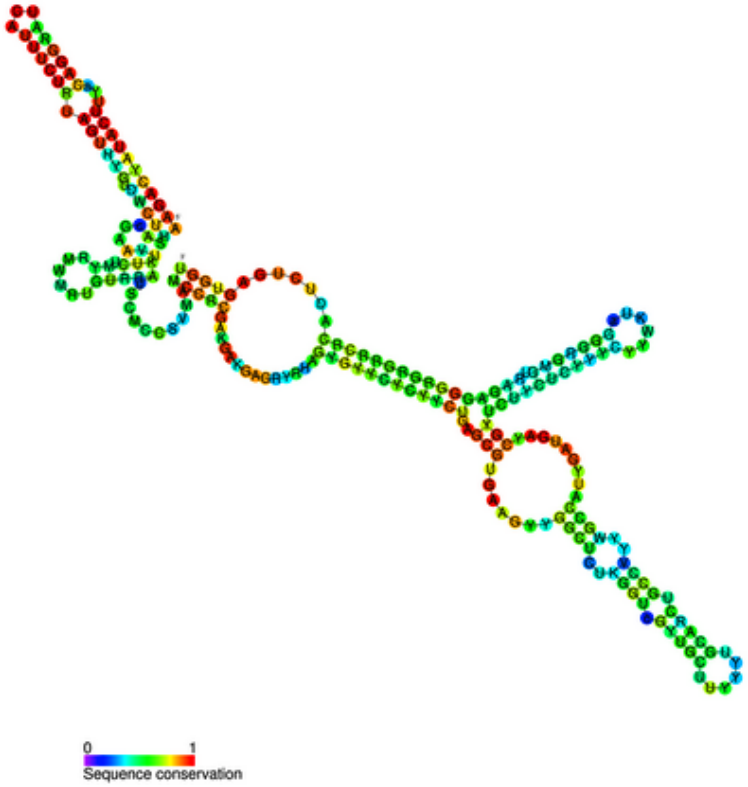


Figura 2.7: Estrutura secundária do U3 snoRNA [26].

O estudo de *ncRNAs* vem se mostrando bastante desafiador devido a dificuldade de se verificar experimentalmente qual função desempenha determinado gene não-codificador de proteínas em um organismo [20]. Métodos computacionais para a identificação de ncRNAs possuem dificuldades similares aos métodos experimentais, e sendo que a Bioinformática não possui um método único para a identificação e classificação de ncRNAs, embora alguns critérios sejam usados, tais como o fato de *ncRNAs* não possuírem ORFs (*Open Read Frame* ou janela aberta de leitura) longas e que *ncRNAs* têm uma conservação em nível de estrutura secundária, e não primária, o que inviabiliza o uso de métodos computacionais utilizados com sucesso para comparação entre sequências de DNA que codificam proteínas [6]. A figura 2.7 mostra um exemplo de estrutura secundária de um U3 snRNA, um tipo de *ncRNA*.

2.2.1 Classificação de *ncRNAs*

A classificação de *ncRNAs* em diferentes famílias é feita normalmente pela função que esta molécula exerce em determinado organismo [20]. A função de um ncRNA está intimamente ligado à sua estrutura secundária e não à sua estrutura primária [47], logo sua função não pode ser descoberta apenas utilizando similaridades entre sequências.

Segundo Lima [51], as famílias de *ncRNAs* conhecidas até os anos 80 eram apenas as de *tRNAs* (responsáveis pelo transporte de um aminoácido que será usado na síntese de proteína) e as de *rRNA* (RNAs ribossomais responsáveis pela catálise de síntese protéica [15]). Atualmente o número de famílias conhecidas chegam a 2208 segundo dados

disponíveis no banco de dados Rfam [11]. Algumas das principais famílias de ncRNAs são descritas na tabela 2.1.

Tabela 2.1: Algumas famílias importantes de *ncRNA* [6, 20, 51, 53, 61].

Sigla	Nome	Função
tRNA	RNA transportador	Transporte de aminoácidos para a síntese de proteínas
rRNA	RNA ribossomal	Envolvidos com a síntese de proteínas
snoRNA	*Small nucleolar RNA*	Realiza modificações nos rRNAs,tRNAs e snRNAs
snRNA	*Small nuclear RNA*	Envolvidos na excisão dos introns no processo de *splicing*
siRNA	*Small interfering RNA*	Interfere na tradução de proteínas separando e promovendo a degradação de trechos de mRNAs
rasiRNA	*Repeat-associated siRNA*	Silenciamento da transcrição de genes via remodelagem da cromatina
snmRNA	*Small non-messenger RNA*	Pequenos *ncRNAs* com função regulatória
miRNA	*MicroRNA*	Família de genes reguladores da tradução
lncRNA	*ncRNAs* longos	Diversas funcionalidades das quais muitas ainda são desconhecidas, como a regulação da expressão gênica a nível de remodelagem de cromatina (possuem mais de 200 nucleotídeos)
piRNA	*Piwi-interacting RNA*	Regulação de tradução e estabilidade de mRNA, dentre outras funções
stRNA	*Small temporal RNA*	Interrompem a tradução de mRNA

Estima-se que existam muitas outras famílias de *ncRNAs* ainda não descobertas [23, 34, 51].

2.2.2 Bancos de dados de *ncRNAs*

Várias classes diferentes de ncRNAs vêm sendo descobertas recentemente [11], implicando no aumento da quantidade de dados sobre essas moléculas. Os bancos de dados de *ncRNAs* têm como objetivo organizar informações relevantes sobre os diversos tipos de *ncRNAs* existentes [69], sendo utilizados tanto para pesquisas por biólogos quanto para identificar novas sequências de ncRNAs em métodos computacionais como aprendizagem de máquina. Em seguida, listamos e comentamos brevemente bancos de dados de *ncRNAs*:

- **ncRNAdb** [69] (*noncoding RNA database*) tem por objetivo prover informação de sequências e funções dos *ncRNAs*. Atualmente este banco de dados inclui sequências de *ncRNAs* provenientes de 99 espécies de bactérias, Archaea e eucariotos.
- **NONCODE** [49] é um banco de dados de *ncRNAs*, extraídos automaticamente da literatura e do GenBank [8], os quais foram manualmente curados. O NONCODE possui quase todos os tipos de *ncRNAs* com exceção de *tRNAs* e de *rRNAs* e todas as suas sequências e informações relacionadas (como função, localização celular, dentre outros) foram confirmadas manualmente. Mais de 80% de suas entradas são baseadas em dados experimentais.
- **RFAM** [11] categoriza sequências primárias de *ncRNA* e estruturas secundárias através do uso de alinhamento múltiplo, consenso de anotações de estruturas secundárias e modelos de covariância. Na sua primeira publicação, o RFAM v1.0 possuía 25 famílias de *ncRNAs* e na sua atual versão v11.0 possui 2.208 famílias.

- **RNAdb** [59] é um banco de dados de *ncRNAs* de mamíferos contendo sequências de nucleotídeos e anotações de dezenas de milhares de *no-housekeeping ncRNAs*, incluindo uma variedade de microRNAs, snoRNAs e grandes *ncRNAs* de tamanho similar ao de um mRNA.
- **fRNAdb** [40] possui uma grande coleção de transcritos não codificadores de proteínas, incluindo sequências anotadas e não anotadas de outros bancos, tais como NONCODE e RNAdb.
- **sno/scaRNAbase** [75] consiste de um banco de dados que possui cerca de 1.979 sno/scaRNA (snoRNAs e *small cajal body-specific RNA*) obtidos de 85 organismos.
- **miRbase** [27] é um repositório de sequências, anotação e predições de microRNAs. Atualmente na sua décima versão ele conta com 5.071 *loci* de *miRNAs* provenientes de 58 espécies, expressando 5.922 miRNAs maduros distintos e, provendo desta forma, uma facilidade para estudo sobre *miRNA*.

2.2.3 Métodos para a identificação de *ncRNAs*

Devido à grande importância biológica dos *ncRNAs* e a grande dificuldade de identificação e classificação, diversos métodos computacionais foram e estão sendo desenvolvidos para a identificação e classificação de *ncRNAs*. Contudo, como já citado anteriormente, a função de um ncRNA está intimamente ligado à sua estrutura secundária e não à sua estrutura primária [47] dificultando a predição deste tipo de molécula com métodos normalmente projetados para identificação de genes codificadores de proteínas [51].

A tarefa de percorrer um genoma com a finalidade de detectar *ncRNAs* é um desafio em aberto na Bioinformática [52]. Estratégias normalmente utilizadas para solucionar este problema são a de identificar atributos e características específicas de famílias de *ncRNAs* [20], ou a de criar programas que podem ser treinados com objetivo de identificar características intrínsecas de uma família específica a partir de uma sequência de entrada [52]. Estas estratégias normalmente seguem processos *ab initio*, isto é, não utilizam outros *ncRNAs* conhecidos a fim de identificar novos, ou métodos comparativos onde a partir de um banco de dados contendo diversas amostras de *ncRNAs* são identificados novos *ncRNAs* classificados em determinada família de *ncRNA*. A seguir são apresentados quatro abordagens comumente utilizadas para a construção de métodos para a identificação de *ncRNAs*.

Abordagem termodinâmica

A sequência de nucleotídeos de uma molécula de RNA carrega informações requeridas para sua conformação em um espaço tridimensional. Estruturas secundárias de RNA podem ser preditas aplicando regras termodinâmicas (energia livre) e topológicas para identificar a estrutura energeticamente mais favorável para uma dada sequência [78]. RNAs que possuem estrutura secundária bem definida têm uma baixa energia livre comparada a sequências com a mesma frequência de nucleotídeos, contudo sem sua estrutura secundária bem definida [20, 52]. Esta característica pode então ser utilizada para detectar *ncRNAs* putativos deslocando uma janela de leitura sobre o genoma e comparando

sua energia livre com versões embaralhadas da mesma sequência [52]. O programa snoReport [30] utiliza o programa RNAfold [31] para o cálculo de energia livre mínima, da qual é utilizada como atributo para a identificação de *snoRNAs*.

Abordagem composicional

Utilizando métodos estatísticos, é possível observar que *ncRNAs* possuem em geral uma média de conteúdo G+C de 50% [63]. Por exemplo, no genoma humano, que possui cerca de 42% de conteúdo GC, *miRNAs* e *snoRNAs* H/ACA box possuem 50% em média de conteúdo GC [72]. Outro fato interessante é o de que em organismos com genomas ricos em dinucleotídeos AT, a busca de *ncRNAs* em regiões ricas em GC é bastante satisfatória [42]. No snoReport [30] o conteúdo GC é utilizado como atributo para a identificação de *snoRNA*.

Abordagem por homologia

A predição de *ncRNAs* é realizada através da comparação de genomas entre duas ou mais espécies. Essas comparações necessitam de bancos de dados curados, pois o quão melhores forem as anotações no banco, melhores serão as predições [52]. Dois genes são ditos homólogos se descendem de um ancestral comum, e possivelmente esses genes possuirão a mesma funcionalidade herdada [6]. Uma ferramenta bastante famosa nesta abordagem é o Infernal [57], que realiza um alinhamento múltiplo considerando a estrutura secundária das sequências [6].

Abordagem utilizando Aprendizagem de Máquina

A Aprendizagem de Máquina vem se mostrando bastante eficaz na construção de programas para identificar *ncRNAs* [52]. Essa abordagem é normalmente utilizada construindo um modelo de gene treinado para identificar determinadas famílias de *ncRNA*. Este modelo de gene normalmente é construído usando atributos de transcritos de ncRNA e mRNA, por exemplo tamanho da ORF, composição de nucleotídeos, estrutura secundária, dentre outros [20], extraídos de diversas amostras disponíveis nos bancos de dados de *ncRNAs*.

2.3 *snoRNAs*

Nesta seção, inicialmente descrevemos as características de *snoRNAs* e em seguida o snoReport, a ferramenta desenvolvida por Hertel et al [30] para identificação de snoRNAs.

2.3.1 Descrição

Como já discutido anteriormente, a maior parte do genoma de organismos mais complexos é transcrito em *ncRNAs* [20]. Alguns *ncRNAs* recém-gerados na transcrição sofrem um processamento posterior, gerando RNAs menores e metabolicamente estáveis possuindo diversas funções, como os *rRNAs, tRNAs* e *snRNAs*. Os *ncRNAs* gerados por esse pós-processamento são então modificados por um outro *ncRNA* denominado *snoRNA* [24].

Os *snoRNAs* são RNAs que possuem de 60 a 300 nucleotídeos que se acumulam no nucléolo[1]. Os snoRNAs são classificados de acordo com determinadas características presentes em sua sequência secundária e possuem duas classes principais: *snoRNAs* C/D box e *snoRNAs* H/ACA box. Em humanos, estes RNAs são normalmente encontrados em regiões intrônicas. Após o processo de *splicing*, os introns são removidos em forma de laços que são abertos e subsequentemente degradados. Contudo os *snoRNAs* escapam desta degradação formando um complexo protéico [24].

Complexos protéicos contendo *snoRNAs*, denominadas *snRNPs* (ribonucleo proteína de pequenos nucleolares) contém proteínas com atividades enzimáticas. No caso de *snoRNAs* do tipo C/D box, sua composição possui fibrilarina que promove a *2'-O-metilação* de seu RNA alvo. Já os snoRNAs do tipo H/ACA box, apresentam em sua composição *disquerina* que catalisa a conversão de uridina para pseudouridina [24].

Os *snoRNAs* C/D box (figura 2.8) são caracterizados pela presença de dois motivos conservados, o box C (RUGAUGA) e o box D (CUGA) encontrados perto das extremidades das fitas 5' e 3' da molécula, respectivamente. Um segundo par de boxes, C' e D', podem ser normalmente encontrados próximos do meio de um *snoRNA* C/D box, mas mostra baixa conservação de sequência quando comparados ao boxes C e D. A região guia (também chamada de box antisenso), trecho do *snoRNA* C/D box complementar a algum RNA alvo, é localizada na linha 5' ao encontro de regiões box D ou D' [67].

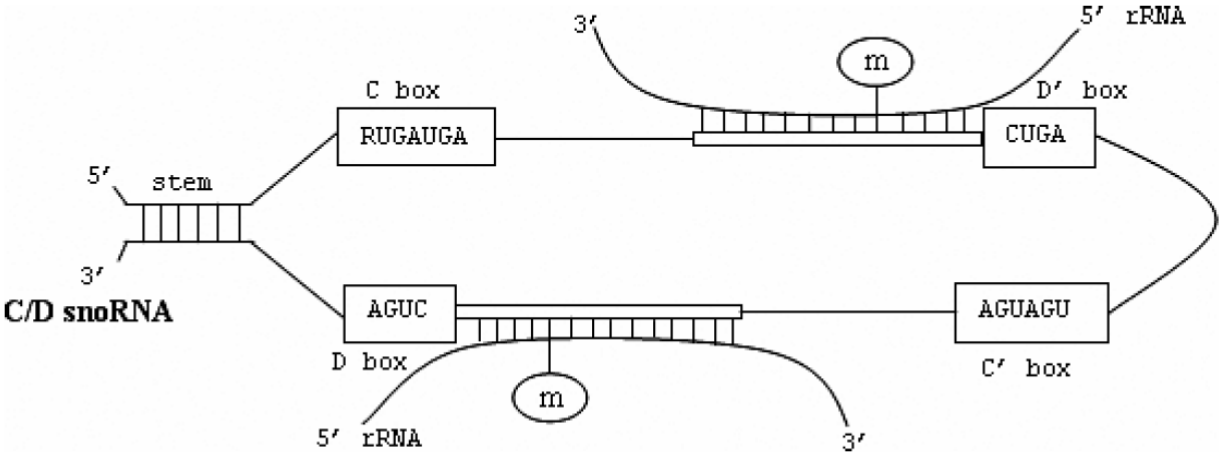


Figura 2.8: Estrutura secundária canônica de um *snoRNA* C/D box. [76]. Os boxes C e D são unidos por um pequeno talo (*stem*) terminal (4-5 pb) e toda região entre esses dois boxes permanecem não pareadas

Já os *snoRNAs* H/ACA box (figura 2.9) são caracterizados por duas estruturas imperfeitas de *hairpins* separados por uma fita contendo o box H (ANANNA) e seguida por uma pequena sequência contendo o motivo ACA localizado 3 nucleotídeos antes do fim da fita 3' [45].

Um grupo de *snoRNAs* que vem sendo descrito em eucariotos de maior complexidade, são os denominados *snoRNAs* órfãos, os quais possuem a característica de não terem

[1]nucléolo: local dentro do núcleo celular em células de eucariotos, onde o *rRNA* é sintetizado [16, 19].

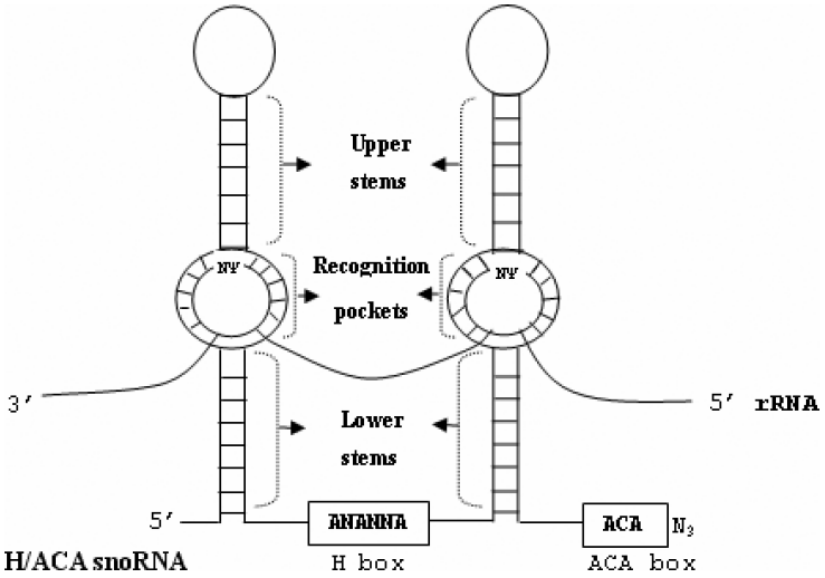


Figura 2.9: Estrutura secundária canônica de um H/ACA box snoRNA [76]. A estrutura secundária de um *snoRNA* H/ACA box consiste de dois *hairpins* e duas pequenas regiões de fita simples, contendo o box H e o box ACA. Os hairpins contem *bulges*, ou *loops* de reconhecimento, que formam um complexo de *pseudoknot* com o RNA alvo. Pseudoknot é uma estrutura secundária de ácidos nucléicos contendo ao menos dois *hairpin-loops* (*stem-loops*), na qual a metade de um *stem* é intercalada entre as duas metades do outro *stem*

complementaridade com *rRNAs*, *tRNAs* e *snRNAs* [30, 39]. Muitos *snoRNAs* órfãos possuem complementaridade com *mRNAs*, sugerindo outras funções [39]. Por exemplo, o *snoRNA* C/D box SNORD115 (HBII-52) possui complementaridade com o receptor 2C de serotonina do *pré-mRNA*, o que influencia o *splicing alternativo* desse *pré-mRNA* [24, 41]. Além disso, alguns *snoRNAs* canônicos (com *RNAs* alvo conhecidos) foram identificados acumulando-se no citosol em condições de estresse celular, dando a possibilidade de inferir que snoRNAs podem ter funções fora do nucléolo [12, 24].

2.3.2 snoReport

A construção de métodos computacionais capazes de identificar as duas classes de *snoRNAs* (C/D box e H/ACA *snoRNA*) vem mostrando ser um problema bastante desafiador [30]. A maioria destes métodos utiliza o fato de que os snoRNAs possuem um pequeno trecho de sequência, chamado box antisenso, que é complementar a um RNA alvo, sendo que tal informação pode aumentar dramaticamente a sensibilidade e especificidade de seus algoritmos [30, 64, 67].

Um método computacional que vem mostrando bons resultados é o snoReport [30], que combina predição de estrutura secundária de *snoRNAs* com Máquina de Vetores de Suporte (SVM), e esta é capaz de reconhecer *snoRNAs* em sequências individuais sem incluir informação sobre o RNA alvo complementar ao bloco antisenso. Embora esta informação possa melhorar consideravelmente a sensibilidade e especificidade, um

número crescente de *snoRNA* órfãos estão sendo descobertos com a característica da falta de complementaridade do box antisenso com o RNA alvo [35, 36], como o caso de um subgrupo de *snoRNA* expresso no cérebro de mamíferos, que parece não estar envolvido no processo de modificação de *rRNAs* e *snRNAs* [71].

Descrição do método

Como o snoReport utiliza métodos de aprendizagem de máquina, o conjunto de treinamento consiste de amostras positivas, sequências das duas principais classes de *snoRNAS*, e amostras negativas, um conjunto de sequências que não são *snoRNAs*. Neste caso o conjunto de amostras positivas frequentemente é pequeno. Hertel [30] usa como amostras positivas *ncRNAs* retirados do banco de dados snoRNABase [45], enquanto o conjunto de amostras negativas foram retirados do Rfam [11], do miRBase [27] e de um conjunto aleatório de sequências. Agrupamentos (ou *clusters*) de sequências homólogas foram determinadas utilizando blastclust [4] com um limiar *E-value* de 10^{-3}. Para cada *cluster*, apenas uma sequência é mantida. Deste modo, evita-se correlações entre os conjuntos de teste e de treinamento nos experimentos de validação [30].

Matrizes de pesos para posições específicas (*PWMs*) foram utilizadas para representar as sequências características dos motivos (*motifs*). Essas foram extraídas do banco snoRNABase usando a ferramenta web meme [7]. A figura 2.10 mostra os motivos resultantes.

(a) Box H (b) Box C (c) Box D

Figura 2.10: Gráfico de frequência dos motivos dos boxes H, C e D. O box ACA foi omitido pois possui exatamente o padrão ACA em seu gráfico de frequência [30].

Componentes de Software

A predição da estrutura do RNA é realizada através do pacote *Vienna RNA Package 1.6* [31, 32]. Já os *PWMs* são comparados à sequência genômica usando pwmatch [56]. Para a implementação da SVM foi utilizada a libsvm 2.83 [13]. O programa snoReport foi escrito na linguagem C ANSI e usa todos os softwares aqui citados.

Workflow do snoReport

O objetivo do snoReport é distinguir *snoRNAs* de outros tipos de *ncRNAs*, em particular reconhecer, de forma confiável, os membros das duas classes principais de *snoRNAs*. Ambas as classes de *snoRNAs*, tanto de classe C/D box quanto de H/ACA box, são distintos, por suas características de suas estruturas secundárias. Por esse fato, a primeira ação realizada pelo snoReport é a predição da estrutura secundária de uma sequência.

Para ambas as classes é definida uma estrutura protótipo com restrições utilizadas na fase de predição da estrutura secundária. As restrições para essa estrutura são:

- Os boxes devem estar localizados em regiões não pareadas;

- A distância entre o box C ao box D e do box H ao box ACA, devem possuir uma distância máxima de 200 e 120 nt, respectivamente;
- Os boxes C e D devem estar de 3 a 15 nt distantes das extremidades da fita;
- Toda a região entre os boxes C e D devem ser não pareados;
- O box H deve estar de 40 a 120 nucleotídeos de distância da extremidade da fita 5';
- O box ACA deve estar na extremidade da fita 3';
- Descarte os candidatos a *snoRNA* com *matching scores* de seus boxes abaixo de 0.5.

Na figura 2.11 é possível visualizar as restrições relacionadas às distâncias entre boxes e das extremidades da fita. Com essas restrições o tempo de execução do snoReport é reduzido, pois os candidados que não se adequarem a elas são descartados.

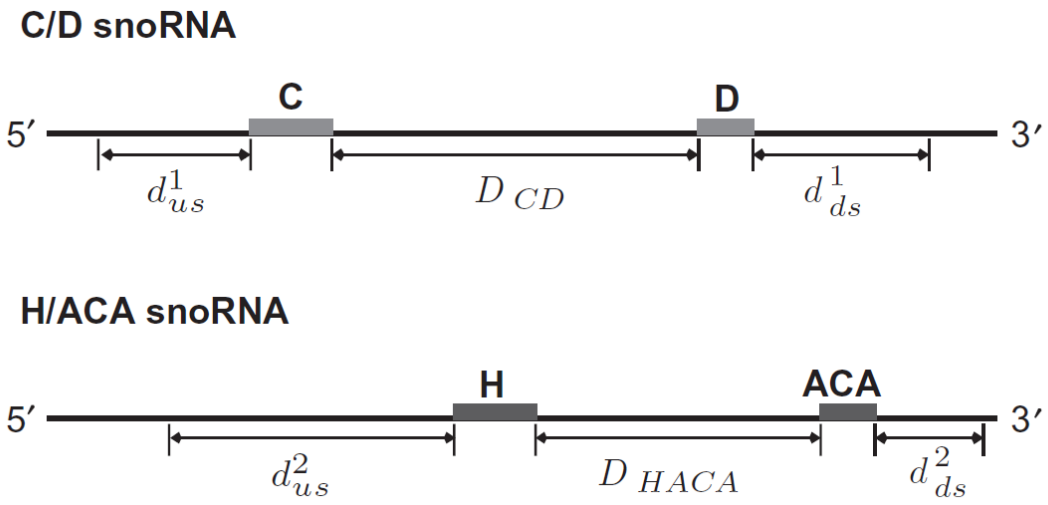


Figura 2.11: Definição dos limites de um *snoRNA* com respeito aos boxes encontrados na sequência de entrada. As regiões especificadas são: $d^1_{us}, d^1_{ds} \in [3, 15]$ nt e $D_{CD} \in [3, 200]$ para C/D box snoRNA, e d^2_{us} tão como $D_{HACA} \in [40, 120]$ nt e $d^2_{ds} = 0$ para o box H/ACA *snoRNA* [30]

Usando estas restrições descritas acima, a sequência dobra-se em uma estrutura protótipo de *snoRNA*, pelo RNAfold [31], e o vetor de características da sequência/estrutura é computado e passado para a SVM utilizada no snoReport. Uma opção adicional no snoReport é a de permitir uma varredura no complemento reverso da sequência. Um candidato é classificado como positivo, isto é, um candidato a snoRNA putativo, se a classificação da SVM retornar uma classificação $pSVM > 0.5$. Caso o candidato tenha $pSVM > 0.9$ é considerado como um candidato de alto *score*.

Na figura 2.12 é possível visualizar o *workflow* utilizado para a identificação das duas principais classes de *snoRNAs*, *snoRNA* C/D box e *snoRNA* H/ACA box. É importante notar que para cada classe de *snoRNA* é realizado o mesmo *workflow*, mas com dados distintos e mudanças no vetor de características passado ao classificador SVM. Os vetores de características de cada classe serão explicados a seguir.

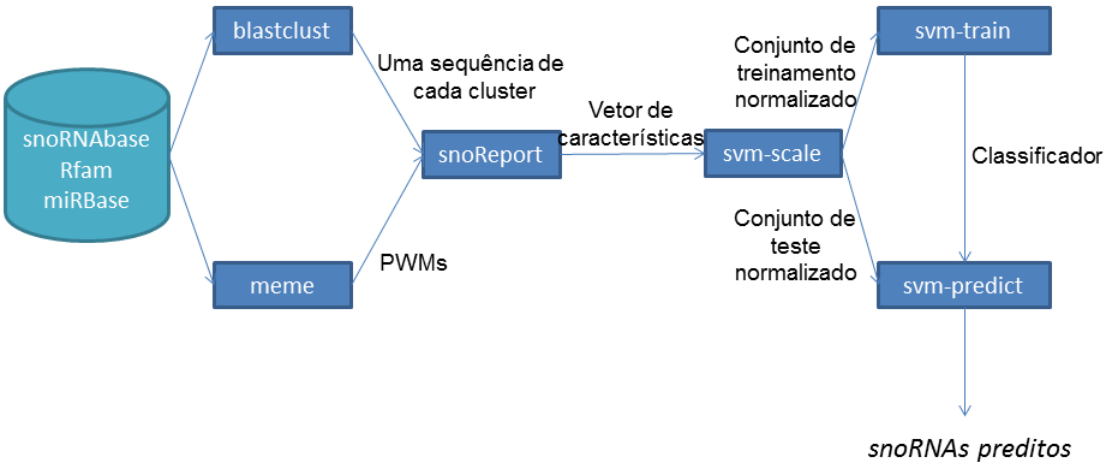


Figura 2.12: *Workflow* usado pelo snoReport. As sequências do conjunto positivo e negativo para ambas as classes de *snoRNA* (*snoRNA* C/D box e *snoRNA* H/ACA box) são retiradas dos bancos de dados snoRNAbase, Rfam e miRBase. Em seguida o blastclust é utilizado para retirar sequências redundantes e, por meio ferramenta meme são obtidas as PWMs. O snoReport recebe então as sequências não redundantes e as PWMs e retorna um vetor de características para cada sequência computada. Depois disso o vetor de características é normalizado pelo programa svm-scale e dividido em um conjunto de treinamento e de teste. O programa svm-train recebe como entrada o conjunto de treinamento normalizado e é gerado um classificador utilizado pelo programa svm-predict para predizer *snoRNAs* a partir de um conjunto de teste.

snoRNA H/ACA box

O box ACA está localizado imediatamente depois da última base pareada do segundo *hairpin*. Usando o fato que ambos os boxes devem ser não pareados, a estrutura secundária da sequência é computada. Após isso, é realizada uma verificação da estrutura secundária do candidato a *snoRNA* H/ACA box, onde sua estrutura deve apresentar um único *hairpin* entre o box H e o box ACA e que haja um *hairpin* precedendo o box H. Após a validação do teste inicial a sequência é truncada novamente na extremidade 5' antes do primeiro par de bases do primeiro *hairpin*. Por fim, a estrutura resultante é recomputada e o primeiro *hairpin* é ajustado, caso necessário. Na tabela 2.2 são mostrados os atributos extraídos da estrutura predita e na sequência truncada utilizados na SVM.

As variáveis E e E_c são utilizadas para medir quanto "esforço" é necessário para forçar que uma sequência adquira a estrutura requisitada (estrutura de um H/ACA box) ou se essa sequência dobra em outra estrutura mais estável (ou não). Os atributos E_{avg} e E_{std} representam a média e desvio padrão da energia livre mínima. Os tamanhos de ambas as hastes e regiões de *hairpin loop* assim como as distâncias D_{H2I5}, D_{A2I3} e D_{HACA} asseguram que seja classificado como *snoRNA* H/ACA box apenas candidatos

Tabela 2.2: Atributos utilizados na SVM extraídos de candidatos a *snoRNA* H/ACA box [30]

E	Energia livre mínima (MFE) sem restrições para a estrutura secundária
E_c	MFE quando há o dobramento com restrições para a estrutura secundária
E_{avg}	MFE média
E_{stdv}	Desvio padrão da MFE
L_{s5}	Número de pares de bases no *stem* 5'
L_{s3}	Número de pares de bases no *stem* 3'
L_{15}	Número de bases não pareadas no *hairpin loop* 5'
L_{13}	Número de bases não pareadas no *hairpin loop* 3'
D_{H2I5}	Distância do box H para a primeira base não pareada do primeiro *loop* interior da haste 5'
D_{A2I3}	Distância do box ACA para a primeira base não pareada do primeiro *loop* interior da haste 3'
D_{HACA}	Distância do box H para o box ACA
GC	Conteúdo GC da subsequência
σ_H	*pwmatch score* do box H

com estruturas semelhantes com essa classe. O conteúdo GC serve como um indicador de estabilidade termodinâmica da estrutura.

snoRNA C/D box

Toda a região, do começo do box C ao fim do box D, deve permanecer não pareado. Este grande *loop* é delimitado por um *stem* curto contendo em média de 4 a 10 pb. Dependendo da posição de inicio i_C do box C e a posição de término j_D do box D na sequência, a sequência é truncada 15 nt acima do box C e abaixo do box D. Se $i_C < 15$ a sequência não é truncada na extremidade 5'. Caso $j_D < T_{3'} - 15$, onde $T_{3'}$ é a posição da extremidade 3', a sequência não será truncada no final da 3'. Além disso, i_C e i_D devem estar pelo menos 3 nt distantes das posições de começo e fim da sequência de entrada.

A sequência restante é então dobrada forçando a região entre os boxes a permanecerem não pareadas. A tabela 2.3 mostra os atributos enviados a SVM de candidatos a *snoRNAs* C/D box.

Resultados da antiga versão do snoReport [30]

No processo de aprendizagem de ambas as classes de *snoRNA*,*snoRNA* H/ACA box e *snoRNA* C/D box, foi utilizada validação cruzada com conjuntos de dados distribuídos aleatoriamente para o conjunto de treinamento e teste. Para este fim os dados positivos e negativos foram particionados em amostras de 50% para treinamento e para teste, 80% treinamento e 20% teste e 90% treinamento e 10% teste, respectivamente.

Os vetores de características do conjunto de treinamento são normalizados na faixa de $[-1, 1]$ antes de ser criado o classificador (que será utilizado para a detecção de *snoRNAs*).

Tabela 2.3: Atributos utilizados na SVM extraídos de candidatos a *snoRNA* C/D box [30]

E	Energia livre mínima (MFE) quando há o dobramento sem restrições
E_c	MFE quando há o dobramento com restrições
E_{avg}	Média da MFE
E_{stdv}	Desvio padrão da MFE
L_s	Número de pares de bases na haste terminal
L_1	Número de bases não pareadas no *hairpin loop*
D_{CD}	Distância do box C para o box D
σ_C	*Matching score* do box C
σ_D	*Matching score* do box D
GC	Conteúdo GC da subsequência

O tipo de SVM utilizado para construir o classificador é o C-SVC, com função *kernel* RBF (*Radial Basis Function*) e $\gamma = 2$ e $C = 1$.

Após a geração do classificador foi gerada a fase de testes, onde o snoReport obteve 96% de sensibilidade e 91% de especificidade para o classificador de *snoRNA* C/D box. Para o classificador de H/ACA *snoRNA* obteve-se 78% de sensibilidade e especificidade de 89%. Em ambos os casos os valores correspondem ao particionamento de 80% dos dados para teste e 20% para treinamento.

Outro teste foi realizado utilizando-se uma região rica em *snoRNAs* no genoma humano (cromossomo 11, posições 93103000-93108500, hg18). Esta região contém 2 *snoRNA* C/D box e 6 *snoRNA* H/ACA box. Para verificar a taxa de falsos positivos, 100 sequências foram criadas, usando essa região do genoma humano, por um procedimento de "embaralhamento"de nucleotídeos utilizando o pacote de ferramentas *SQUID* [22]. Nesse teste, o snoReport identificou todos os *snoRNAs* H/ACA box e *snoRNAs* C/D box com exceção do ACA25 (*H/ACA snoRNA*), e não reportou nenhum candidato adicional, isto é, não classificou erroneamente as sequências criadas a partir do embaralhamento de nucleotídeos. O ACA25 não foi identificado pelo snoReport, pois o box H está localizado numa região pareada do *stem* 5', logo, devido as restrições impostas no snoReport, esta sequência é descartada antes mesmo de ser utilizada no classificador.

Outros testes foram feitos pelo snoReport, a fim de verificar a qualidade das predições. Foi submetido ao snoReport um conjunto de *snoRNAs* preditos e parcialmente verificados em humanos, nematóides, drosófilas e espécies do gênero leishmania. A tabela 2.4 apresenta os resultados encontrados nestes dados. Para os de Leishmania não foram identificados *snoRNA* H/ACA box, pois essa classe de *snoRNA* em espécies do gênero leishmania apresenta diferenças nos box H e ACA, em relação aos boxes H e ACA canônicos encontrados em leveduras e vertebrados [30].

Discussão

O snoReport é uma ferramenta capaz de reconhecer e classificar as duas principais classes de *snoRNAs*, *snoRNA* H/ACA box e *snoRNA* C/D box. Ele utiliza a combinação de predição de estrutura secundária e aprendizagem de máquina utilizando SVM. Diferentemente da maioria das outras ferramentas com o mesmo objetivo, o snoReport não

Tabela 2.4: Resultados do snoReport aplicados a snoRNAs reportados em humanos, nematóides, drosófilas e espécies do gênero leishmania. Na tabela é apresentado o número de candidatos positivamente classificado no snoReport seguido pelo número de candidatos reportados na literatura.

Humano		
Yang et al. [76]	CD: 11/21	HACA: 23/32
snoRNA confirmados	CD: 7/10	HACA: 8/10
Nematóides		
Deng et al. [21]	CD: 16/40	HACA: 31/47
Zemann et al. [77]	CD: 33/77	HACA: 44/57
Huang et al. [35]	CD: 8/17	HACA: 10/16
Drosófilas		
Accardo et al.[3]	CDs confirmados: 11/27 CDs não confirmados: 19/70	
Leishmania		
Liang et al. [46]	CD: 7/62	HACA *A-like*: 0/37

utiliza informação de RNAs alvos como *rRNA* ou *snRNA*. No trabalho de Hertel [30], esta ferramenta utiliza sequências individuais como entrada e seu conjunto de treinamento foi formado quase em sua totalidade por sequências de mamíferos e funcionou satisfatoriamente, possuindo uma sensibilidade na ordem de 50% e uma taxa de falsas descobertas com ordem de magnitude inferior a outras abordagens.

2.3.3 Revisão de literatura

Nesta subseção serão apresentados outros métodos encontrados na literatura para a identificação de *snoRNAs*.

snoSeeker

Assim como o snoReport o pacote snoSeeker [76] é capaz de identificar tanto *snoRNAs* guias (aqueles que modificam *rRNAs*, *tRNAs* e *snRNAs*) quanto *snoRNAs* órfãos (não possuem complementaridade com *RNAs* alvo [30, 76]. O pacote snoSeeker possui dois programas principais: CDseeker e o ACAseeker [76], que tem a função de detectar C/D box ısnoRNAs e H/ACA *snoRNAs*, respectivamente. Utilizando o snoSeeker, Yang [76] realizou uma varredura no genoma humano em busca de *snoRNAs* utilizando sequências de alinhamentos (*Whole-genome alignment* (WGA)) entre o genoma humano e quatro espécies de mamíferos (humano/rato, humano/camundongo, humano/cachorro e humano/vaca) e com seu método a maioria dos snoRNAs conhecidos até então da classe C/D box e H/ACA foram detectados.

snoGPS

O snoGPS [65] é um algoritmo determinístico que faz uma varredura em genomas com a finalidade de identificar *snoRNAs* H/ACA box guias. O programa combina o algoritmo determinístico com um modelo probabilístico que atribui uma pontuação para cada candidato a *snoRNA* H/ACA box. O snoGPS foi testado no genoma da *Saccharomyces cerevisiae*, onde seis *snoRNAs* putativos foram encontrados, dos quais cinco são guias para a formação de pseudouridina em rRNAs. Além disso, 41 das 44 modificações de pseudouridina conhecidas no *rRNA* da *S. cerevisiae* foram ligadas a um *snoRNA* H/ACA box predito.

snoScan

O snoScan [50] é baseado em métodos probabilísticos, originalmente utilizados em reconhecimento de fala e linguística computacional. O snoScan foi utilizado para fazer uma varredura no genoma da *S. cerevisiae* e identificou 22 *snoRNAs* guias de metilação, snR50 a snR71. Experimentos de bancada confirmaram as suas funções de metilação. No total 51 dos 55 locais de metilação no *rRNA* da *S. cerevisiae* foram associados a 41 *snoRNAs* guia diferentes

Capítulo 3

Aprendizagem de Máquina

Aprendizagem de Máquina é uma subárea de estudo da Inteligência Artificial que tem como principal foco a questão de como construir programas de computadores que automaticamente se aprimoram com experiência [54]. Nos últimos anos muitas aplicações de grande sucesso utilizando aprendizagem de máquina foram desenvolvidas, tais como programas de mineração de dados que aprendem a detectar transações fraudulentas de cartão de créditos, veículos autônomos que aprendem a dirigir em estradas sem a interação humana, reconhecimento de fala, identificação e classificação de RNAs não codificadores (como é o caso deste trabalho) e muitos outros [54].

Antes de apresentar algumas abordagens utilizadas em programas de aprendizagem de máquina é necessário definir o que seria aprender no contexto de uma máquina. Segundo T. M. Mitchell [54] Um programa de computador é dito que aprendeu a partir de uma experiência E através de alguma classe de tarefas T e uma medida de performance P, Se sua performance para a tarefa T, medida em P, é aprimorada com a experiência E.

Neste Capítulo serão apresentados duas abordagens utilizadas em Aprendizagem de Máquina: Redes Neurais Artificiais (seção 3.1) e Máquinas de Vetores de suporte (SVM), a qual será utilizada neste trabalho.

3.1 Redes neurais artificiais

3.1.1 Redes neurais reais e artificiais

Uma rede neural artificial é um modelo computacional inspirado no cérebro humano que consiste de elementos de processamento (chamados neurônios) e conexões entre eles possuindo um coeficiente (peso) para cada conexão, constituindo uma estrutura neuronal [38]. Redes neurais artificiais também podem ser denominadas como modelos conexionistas devido as suas principais regras de conexões existentes.

Embora redes neurais artificial possuam similaridades com as redes neurais existentes no cérebro de um ser humano, redes neurais reais são mais complexas e infelizmente muitas de suas funções ainda não foram compreendidas em sua totalidade. Entretanto, no passo em que são descobertas novas funções ou características existentes no cérebro, melhores modelos computacionais serão desenvolvidos e sendo úteis para a resolução de uma infinidade de problemas.

As principais características de uma rede neural real ou artificial são:

- **Aprendizado e adaptação**;
- **Generalização**;
- **Paralelismo massivo**;
- **Robustez**;
- **Armazenamento de informação associativa**;
- **Processamento de informação espaço-temporal**.

Neurônios biológicos

O cérebro humano pode ser dividido em 3 níveis no ponto de vista de processamento de informação:

- **Nível estrutural:** neurônios, regiões de neurônios e conexões entre eles;
- **Nível fisiológico:** o modo em que o cérebro processa informações através de reações químicas e físicas e na transmissão de substâncias;
- **Nível cognitivo:** o modo como os humanos pensam.

Um cérebro humano possui em média cerca de 10^{11} neurônios participando algo em torno de 10^{15} interconexões em caminhos de transmissão [38]. Neurônios são bastante parecidos com outras células do corpo, contudo possuem algumas características singulares, tais como a capacidade de receber, processar e transmitir sinais eletroquímicos por caminhos de transmissão que compõem o sistema de comunicação do cérebro. A figura 3.1 mostra a estrutura típica de um neurônio. Um neurônio é composto de um corpo celular, também denominado soma, e de um ou mais ramos, que podem ser ou dendritos, responsáveis por conduzir a informação para dentro da célula (estímulo) ou axônios, dos quais tem por função conduzir informação para fora da célula (reação).

A interação entre neurônios acontece em determinados pontos de contato denominados sinápses. Uma sinápse pode ser dividida em duas partes: A membrana pré-sináptica, pertencendo ao neurônio transmissor, e a membrana pós-sináptica, pertencendo ao neurônio receptor. Um impulso é emitido do neurônio transmissor enviando da parte pré-sináptica para a pós-sináptica uma substância denominada mediador. Na membrana do neurônio receptor o nível de permeabilidade muda proporcionalmente pela soma algébrica dos potenciais recebidos na parte pós-sináptica. Se o resultado desta soma supera um valor limite estabelecido de antemão, o neurônio terá a função de transmitir uma ação potencial, isto é, ativará outro neurônio.

O cérebro consiste de diferentes neurônios diferindo um ao outro por sua forma. Há uma correlação entre a função da célula e sua forma. Em termos de variedade, mais de 50 tipos de funcionalidades diferentes de neurônios são encontrados no cerebelo. Este fato influencia a criação de modelos computacionais neuronais heterogêneos que consistem de diferentes tipos de neurônios artificiais especializados [38].

O cérebro humano é um sistema de comunicação bastante complexo. Um simples impulso nervoso de um neurônio carrega uma pequena quantidade de informação. Logo o processamento de informações complexas só é possível através da interação de grupos de vários neurônios. A presença de uma grande quantidade ligações entre neurônios

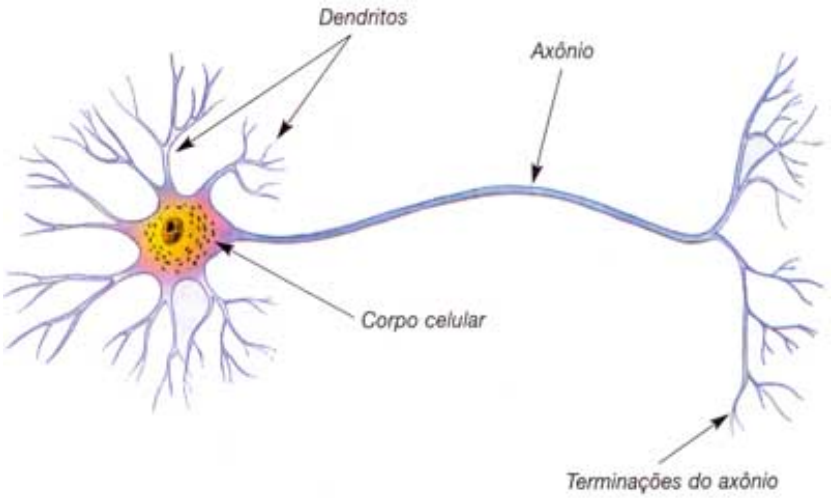


Figura 3.1: Estrutura típica de um neurônio humano. Um neurônio é composto basicamente por dendritos e por um axônio. A interação entre neurônios se dá através das sinápses [10].

determina um paralelismo massivo do cérebro humano. De fato, enquanto a reação de um neurônio a um determinado estímulo seja razoavelmente devagar (algo em torno de 200ms [38]), o cérebro humano como um todo é capaz de resolver problemas complexos em um tempo comparativamente baixo em relação, por exemplo, a um sistema de computação.

A característica do cérebro em analisar problemas complexos e de reagir adequadamente em situações não familiares se dá pela sua habilidade heurística de tomar decisões com base em conhecimentos anteriores e sua habilidade em se adaptar a novas situações. Nosso cérebro possui a habilidade de aprender e de generalizar. A informação acumulada como resultado de nosso aprendizado é armazenada nas sinápses na forma de substâncias químicas concentradas.

A simulação de um cérebro humano é um grande desafio para o século XXI e projetos de grande porte como o *Human Brain Project* [2] estão juntando esforços a nível global para este problema. Mesmo sendo uma tarefa bastante difícil, muitos modelos e sistemas computacionais podem ser desenvolvidos baseados nos princípios e características observadas no cérebro humano para a resolução de uma grande gama de problemas.

Neurônios artificiais

O primeiro modelo matemático de um neurônio foi proposto por McCulloch e Pitts em 1943. Tratava-se de um dispositivo binário usando entrada binária, saída binária e um limiar de ativação fixo. Em geral um modelo de um neurônio artificial (figura 3.2) é baseado nos seguintes parâmetros:

- Conexões de entrada (ou entradas): $x_1, x_2, ..x_n$. Também há em cada conexão de entrada um peso atribuído: $w_1, w_2, ..., w_n$. Uma das entradas do neurônio, chamada de viés, possui um valor constante 1 e é normalmente representado como a entrada x_0;

- Uma função de entrada f que calcula o agregado da entrada do neurônio $u = f(x_i, w_i)$ onde x_i é a i-ésima entrada e w_i o i-ésimo peso referente a esta entrada. f é normalmente uma função somatório: $u = \sum_{i=1}^{n} x_i.w_i$;
- Uma função de ativação s que calcula o nível de ativação de um neurônio $a = s(u)$;
- Uma função de saída que calcula o valor do sinal de saída emitido pela saída (axônio) do neurônio: $o = g(a)$. O sinal de saída é normalmente assumido ser igual ao nível de ativação do neurônio, isto é, $o = a$.

De acordo com os valores que cada parâmetro pode receber, diferentes tipos de neurônios podem ser construídos. Os valores de entrada e saída podem ser binários, {0,1}; bivalentes, {-1,1}; contínuos, [0,1]; ou números discretos em um intervalo definido.

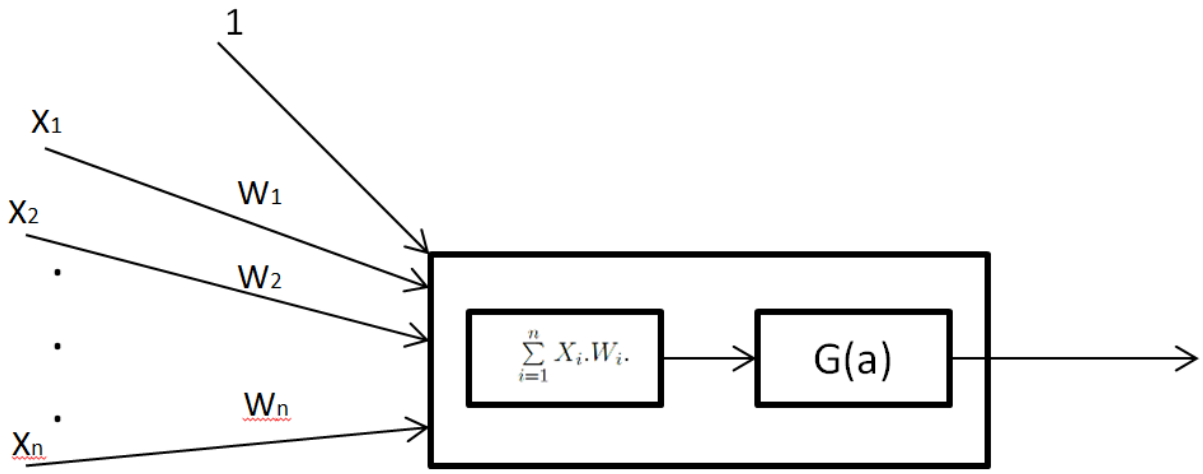


Figura 3.2: Modelo de um neurônio artificial.

De neurônios para redes neurais

Embora um simples neurônio pode atuar em simples funções de processamento de funções, o poder de uma computação neuronal vem em conectar vários neurônios em uma rede. O primeiro modelo operacional de neurônio artificial proposto por Rosenblatt (1958), denominado como *perceptron*, resolvia diversos problemas, contudo em 1969 Minsky e Papert demonstraram limites teóricos no perceptron. Com isso muitos pesquisadores abandonaram o estudo de redes neurais. Contudo novos modelos conexionistas, tais como **memórias associativas , perceptron de multicamadas, algoritmos de** *backpropagation*, **dentre outras** foram desenvolvidos trazendo de volta o interesse dos pesquisadores para a área de Redes Neurais.

Uma rede neural artificial é um modelo computacional definido por quatro parâmetros:

1. Tipo de neurônios (também chamados de nós, considerando uma rede neural um grafo);
2. Arquitetura conexionista - A organização das conexões entre neurônios;

3. Algoritmo de aprendizado;
4. Algoritmo de *recall*.

Um exemplo de rede neural retirado do livro de N.K. Kasabov [38] pode ser visto na figura 3.3. Esta rede contém quatro nós de entrada, dois intermediários e um nó de saída. Os pesos das conexões são configurados como resultado de um treinamento advindo de um algoritmo de aprendizado. Uma nova amostra, por exemplo o vetor (1,0,1,0), percorrerá a rede neural e terá como saída o sinal 1 quando o limiar da função de ativação possui um valor limiar de zero, que é o caso deste exemplo. Em nosso exemplo vemos por exemplo que no nó n5 o resultado da função de entrada é negativo e logo não supera o limiar de ativação.

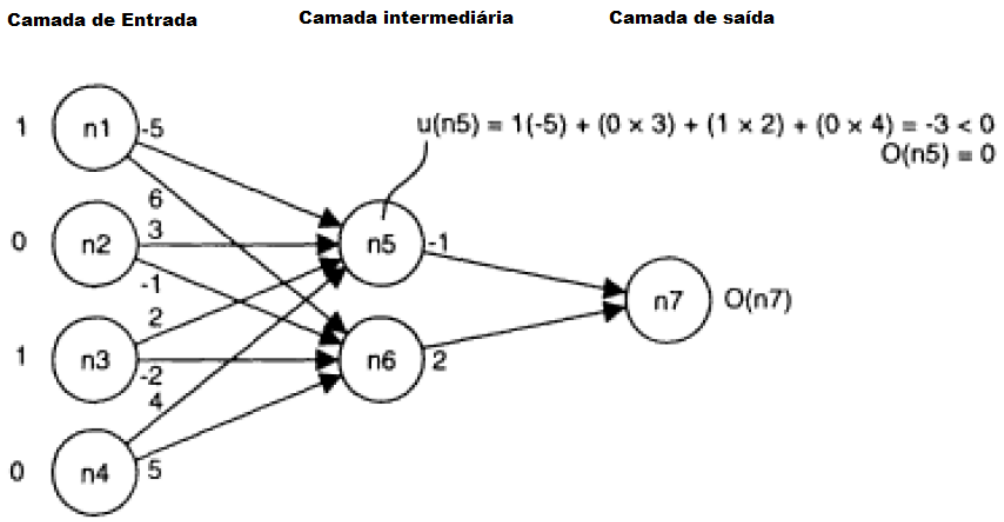


Figura 3.3: Exemplo de uma rede neural contendo 4 nós de entrada, dois nós intermediários e um nó de saída. Os pesos de cada conexão são resultados de treinamento feito anteriormente na rede. O cálculo da função de entrada e o valor limiar da função de ativação do neurônio n5 é também mostrado na figura. [29].

O funcionamento de uma rede neural, quando um vetor de entrada x é submetido, pode ser visualizado como uma função de mapeamento $F : X \rightarrow Y$ onde X é o conjunto de entrada e Y é o conjunto de saída da rede neural. Logo um vetor x $\in$ X é mapeado para um vetor de saída y $\in$ Y através de um "filtro"de pesos, isto é, $y = F(x) = s(w, x)$, onde w é uma matriz de pesos. A matriz de pesos representa o "conhecimento", a memória de longo prazo do sistema enquanto a ativação dos neurônios representa o estado atual, isto é a memória de curto prazo.

O tipo de conexões entre neurônios em uma rede neural define a sua topologia. Neurônios pertencentes a uma rede neural podem ser totalmente conectados, isto é, todo neurônio é conectado a todos os outros, ou parcialmente conectados, como por exemplo apenas conexões entre diferentes camadas são permitidas ou nem todas conexões entre neurônios são existentes.

As duas principais arquiteturas conexionistas podem ser distinguidas através do números de neurônios de entrada e saída e pelas camadas de neurônios usadas: auto-associativas, onde os neurônios de entrada são também o de saída, como é o caso da **rede Hopfield**, por exemplo; e hétero-associativas, onde há conjuntos separados de neurônios de entrada e neurônios de saída, como é o caso da **rede perceptron multicamada**, por exemplo. A figura 3.4 exemplifica estas arquiteturas.

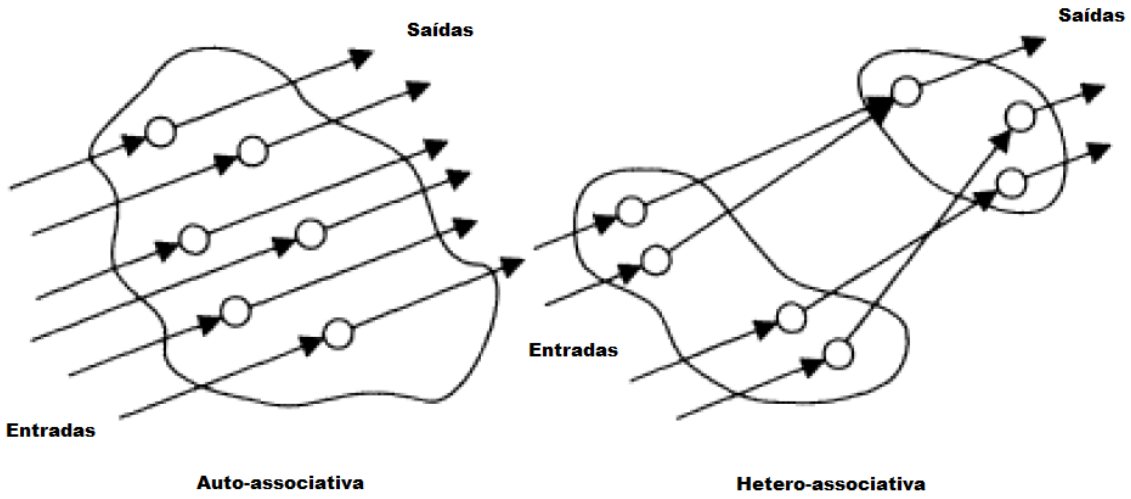


Figura 3.4: Duas principais arquiteturas de redes neurais: auto-associativa (figura da esquerda), e hétero-associativa (figura da direita) [29].

De acordo com a ausência ou presença de conexões de realimentarão, ou em inglês*feedback*, dois tipos de arquiteturas:

- **Arquitetura não-recorrente** (*feedfoward*): Não há conexões voltando dos neurônios de saída para os neurônios de entrada, isto é, a rede não mantém uma memória de valores de saída e dos estados de ativação de seus neurônios em situações passadas;
- **Arquitetura recorrente** (*feedback*): Há conexões voltando dos neurônios de saída para os neurônios de entrada, isto é, a rede neural mantém uma memória de estados passados, e o próximo estado da rede depende não apenas do sinal de entrada mas de estados anteriores da rede neural.

3.1.2 Aprendizagem de Máquina

A habilidade de aprender é uma das mais atrativas características em uma rede neural. O aprendizado é possível devido a modificação do comportamento da rede em resposta a estímulos do ambiente. Uma rede neural é treinada a partir de um conjunto X de vetores de entrada que produzem conjunto Y de vetores de saída ou a rede aprende sobre

características internas e estrutura dos dados do conjunto X. O conjunto X é chamado de *conjunto de treinamento* e seus elementos são denominados amostras de treinamento.

O processo de treinamento se dá em alterar os pesos das conexões na rede neural. Durante o treinamento, os pesos da rede vão gradualmente convergindo a valores tais que para cada vetor x de entrada o vetor y de saída é produzido pela rede neural. O aprendizado ocorre se houver a mudança de ao menos um peso de alguma conexão da rede é alterado a partir das amostras de treinamento.

Uma rede neural alcança um aprendizado aplicando um algoritmo de aprendizagem (também chamado de algoritmo de treinamento). Algoritmos de aprendizagem de máquina podem ser classificados em quatro grupos: **supervisionado**, onde cada amostra de treinamento possui tanto um vetor de entrada quanto um vetor de saída e o aprendizado ocorre ao associar vetor de entrada ao vetor de saída; **não supervisionado**, onde as amostras de treinamento só possuem o vetor de entrada tendo como meta aprender algumas características internas de todo o conjunto de vetores de entrada adquiridos nas amostras; **semi-supervisionado**, neste caso nem todas as amostras do conjunto de treinamento possuem um vetor de saída; e **aprendizagem por reforço**, é uma combinação de aprendizado supervisionado e não supervisionado e é baseado em apresentar um vetor x de entrada para a rede neural e observar o vetor de saída gerado, se é considerado boa a saída, é então dado uma recompensa no sentido em que os pesos das conexões existentes são aumentados, caso contrário é dada uma punição reduzindo os pesos da rede. Neste trabalho será utilizada a aprendizagem supervisionada que será explanada com mais detalhes na próxima seção.

O aprendizado não se dá apenas pela habilidade de um único neurônio, mas de toda rede neural. Uma das mais conhecidas leis de aprendizagem de modelos conexionistas é a lei de aprendizado de Hebbian. Essa lei diz que a sinápse conectando dois neurônios i e j aumenta seu peso w_{ij} se estes dois neurônios são ativados simultaneamente repetidas vezes por um estímulo de entrada. O peso da sinápse muda Δw_{ij} sendo esta calculada como função do produto de dois valores de ativação a_i e a_j dos neurônios i e j:

$$\Delta w_{ij} = c.a_i.a_j$$

O processo geral da aprendizagem de máquina em uma rede neural é descrito pela sua característica de convergência. A rede neural reage cada vez melhor a cada amostra de treinamento x na medida em que são introduzidas na rede, gerando a cada passo uma saída y desejada. Após a rede parar de aprender de suas amostras de treinamentos, os pesos não mudam mais, isto é $\Delta w_{ij} = 0$ para cada conexão (i, j). Uma rede neural pode parar de aprender por duas razões: a primeira é que a rede aprendeu a partir das amostras de treinamento e a segunda é que a rede se tornou saturada. O teorema de saturação de Grossberg diz que sinais de entrada grandes saturam um neurônio quando ele é sensível para sinais de entrada pequenos, mas, caso um neurônio seja sensível a sinais de entrada grandes, sinais de entradas pequenos são ignorados sendo considerados ruídos.

Pequenos valores aleatórios introduzidos como ruídos durante o processo de aprendizado tendem a aumentar a robustez da performance da rede neural. Neste caso a lei de aprendizado de Hebbian será da forma:

$$\Delta w_{ij} = c.a_i.a_j + n$$

onde n é um sinal de ruído. Quando o ruído é apresentado durante o aprendizado a rede neural alcançará a convergência quando seus pesos estiverem abaixo da magnitude do ruído:

$$\Delta w_{ij} \leq n$$

Outra característica intrínseca em redes neurais é a generalização. Esta característica é presente quando, após uma rede neural ser treinada, um vetor de entrada x' é apresentado a rede e um procedimento de *recall* (de lembrança) é ativado. A rede pode então produzir uma saída y' da qual é similar a saída y_i vinda das amostras de treinamento caso x' é similar ao vetor de entrada x_i. Este princípio nos diz que *estímulos similares causam reações similares*. A generalização pode levar uma grande quantidade de iterações para calcular consecutivos estados da rede, como é o caso de redes recorrentes. Eventualmente as redes alcançam um estado de equilíbrio, que se dá quando a rede não muda seu estado durante as próximas iterações.

3.1.3 Aprendizado supervisionado

O aprendizado supervisionado pode ser visto como uma aproximação de um mapeamento entre um domínio e um espaço de solução do problema: $X \rightarrow Y$, quando amostras do treinamento são da forma (x,y) onde $x \in X$, $y \in Y$, $x = (x_1, x_2, ..., x_n)$ o vetor de entrada e $y = (y_1, y_2, ..., y_m)$ o vetor de saída. Pelo fato de as amostras já possuírem o vetor de saída, dizemos que os dados estão rotulados. A aprendizagem supervisionada neste contexto tem a missão de encontrar uma aproximação F' a partir dos dados rotulados de tal forma que possa ser generalizado novas entradas x.

O aprendizado supervisionado utilizado em redes neurais artificiais normalmente é implementado seguindo a seguinte sequência:

1. Escolha uma apropriada estrutura para a rede neural, tendo, por exemplo, $n + 1$ neurônios de entrada (n o número de variáveis de entrada e 1 para o viés x_0), m neurônios de saída e escolha os valores iniciais dos pesos de cada conexão da rede.
2. Alimente a rede neural com um vetor de entrada x vindo de uma das amostra de treinamento X.
3. Calcule o vetor de saída o produzido pela rede neural.
4. Compare a o vetor saída desejado y (retirado da mesma amostra do vetor de entrada) com o vetor de saída obtido pela rede neural. Se possível avalie o erro gerado.
5. Corrija os pesos de conexão a fim de que da próxima vez que x é apresentado a rede, a saída produzida seja próxima ao valor desejado y.
6. Se necessário repita os passos 2 ao 5 até que a rede alcance o estado de convergência.

O erro de aproximação pode ser calculado de diversas formas, o mais utilizado é o erro instantâneo:

$$Err = (o - y) \text{ ou } Err = |o - y|;$$

o erro quadrático médio:

$$Err = (o - y)^2/2 \ ;$$

a soma de todos os erros sobre todas as saídas da amostras e todas as saídas geradas pela rede neural:

$$Err = (\sum_{k=1}^{p} \sum_{j=1}^{m} (o_j^{(k)} - y_j^{(k)})^2)/(p.m);$$

onde: $o_j^{(k)}$ é o valor da j-ésima saída da rede neural quando a k-ésima amostra de treinamento foi apresentada; $y_j^{(k)}$ é a j-ésima saída desejada da k-ésima amostra de treinamento; p é a quantidade de amostras de treinamento e m é a dimensão do espaço de saída, isto é, a quantidade de variáveis independentes igual ao número de neurônios de saída na rede neural.

Dependendo da métrica de erro a ser usada, há dois tipos de erro. O primeiro é chamado de erro aparente, que estima o quão bom foi o treinamento da rede neural através das amostras de treinamento. O segundo tipo de erro é o erro de teste, que estima o quão bom uma rede neural se generaliza, isto é, o quão bom a rede reage com novas entradas.

O aprendizado supervisionado é um paradigma de aprendizagem bastante útil para vários problemas como classificação ou como em aprendizados de certos comportamentos de determinadas classes padrões.

3.2 Máquina de Vetores de Suporte

Nesta seção será apresentada uma nova categoria de redes *feedfoward* conhecidas como Máquina de Vetores de Suporte, ou *Support Vector Machine* (SVM). Assim como as redes neurais, as *SVMs* podem ser utilizadas tanto para classificação quanto para regressão [29]. Os problemas de classificação consistem em determinar qual a classe em que uma determinada amostra se encaixa, sendo que estas classes assumem valores discretos diferentemente dos problemas de regressão onde as classes assumem valores contínuos.

Uma *SVM* é uma máquina linear que tem como principal ideia, no contextos de problemas de classificação de padrões, construir um hiperplano como superfície de decisão de tal modo que a margem de separação entre amostras positivas e negativas é maximizada [29]. Isto é, através da fase de treinamento duas classes serão separadas a partir de uma função de forma que na fase de testes, isto é, onde dados ainda não classificados são fornecidos a SVM, terão suas classes preditas.

Na figura 3.5 existe um conjunto de classificadores lineares (hiperplanos) separando duas classes distintas de amostras. Contudo apenas um classificador maximiza a margem de separação entre estas classes que pode ser visto com mais detalhes na figura 3.6.

Neste contexto temos uma SVM linear, um classificador que maximiza a margem de separação entre amostras positivas e negativas. Mas por que maximizar a margem de separação? Existem diversas explicações incluindo testes empíricos de performance [37]. Outra justificativa é a de que, caso houvesse um pequeno erro na localização das margens de separação, isto é, se ela não estivesse maximizada, haveria chances de erro de

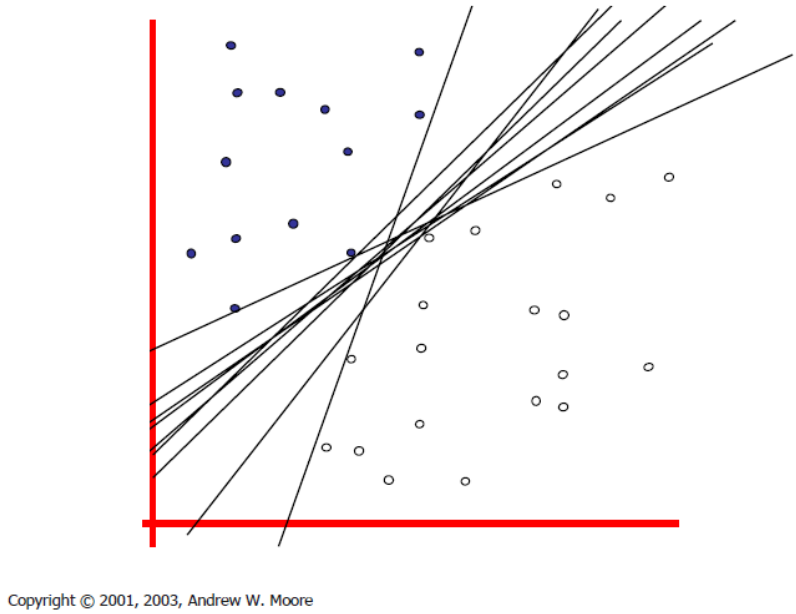


Figura 3.5: Conjunto de classificadores lineares (hiperplanos) separando duas classes distintas de amostras [55]

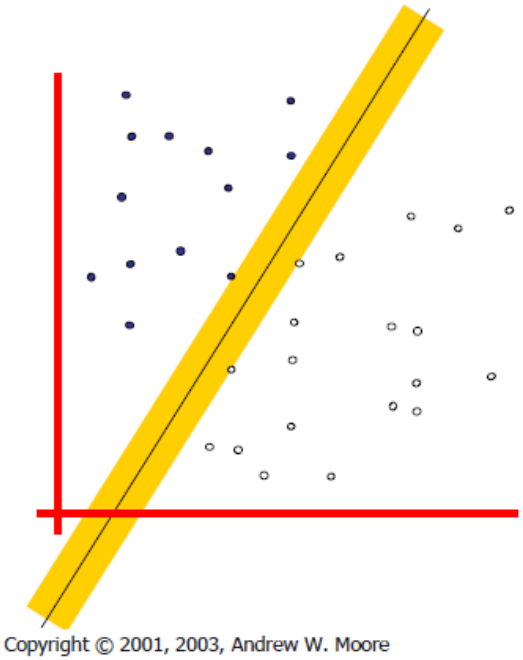


Figura 3.6: O classificador linear que maximiza a margem de separação entre as duas classes é ilustrado apresentando suas margens por um retângulo amarelo. Este é um exemplo de uma SVM linear, também chamada de LSVM [55]

classificação, pois amostras de uma classe, por exemplo positiva, que estariam no interior da margem de separação maximizada no lado do hiperplano das amostras positivas,

poderiam, em uma margem de separação menor, estar no lado das amostras negativas, causando um erro de classificação. Além disto, outra vantagem é a de diminuir a chance de mínimos locais, realizando, assim, uma melhor classificação. [37].

Nas próximas subseções serão apresentadas as principais ideias para a construção de um hiperplano para padrões linearmente separáveis ou não separáveis, também será apresentado soluções a fim de otimizar a localização destes hiperplanos e, por fim, será apresentado uma abordagem a ser utilizada para o uso de SVMs para reconhecimento de padrão.

3.2.1 Hiperplano ótimo para padrões linearmente separáveis

Considere um conjunto de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$, onde x_i é o i-ésimo padrão de entrada e d_i é a saída desejada para essa entrada, ou seja, sua classe. Assuma que as únicas classes válidas para este conjunto de treinamento sejam $d_i = +1$ e $d_i = -1$ e que estas classes são linearmente separáveis. A equação da superfície de decisão que forma um hiperplano que separa estas classes será:

$$w^T x + b = 0 \tag{3.1}$$

onde x é um vetor de entrada, w é um vetor de pesos ajustável, e b é um viés. Sendo assim temos que:

$$w^T x_i + b \geq 0 \quad para \quad d_i = +1 \tag{3.2}$$

$$w^T x_i + b < 0 \quad para \quad d_i = -1 \tag{3.3}$$

Dado um vetor de pesos w e um viés b, a separação entre um hiperplano (definido na equação 3.1) e um ponto referente a uma amostra de treinamento é chamada de margem de separação, denotada por ρ [29]. O objetivo de uma SVM é a de encontrar um hiperplano de tal modo que a margem de separação ρ é maximizada. Sendo assim a superfície de decisão é chamada de hiperplano ótimo [29].

Seja w_0 e b_0 os valores ótimos para o vetor de pesos e do viés, respectivamente. Sendo assim, o hiperplano ótimo, representando uma superfície de decisão linear em um ambiente multidimensional no espaço de entrada é definido por:

$$w_0^T x + b_0 = 0 \tag{3.4}$$

A seguinte função discriminante:

$$g(x) = w_0^T x + b_0 \tag{3.5}$$

nos dá a medida algébrica da distância de x para o hiperplano ótimo [29]. Expressando x como:

$$x = x_p + r\frac{w_0}{||w_0||} \tag{3.6}$$

Onde x_p é a projeção normal de x em direção ao hiperplano e r é a distância algébrica desejada, onde r é positivo se x está no lado positivo do hiperplano e negativo se x está no lado negativo. Sendo que, por definição, $g(x_p) = 0$, segue que:

$$g(x) = w_0^T x + b_0 = r||w_0|| \tag{3.7}$$

ou

$$r = \frac{g(x)}{||w_0||} \tag{3.8}$$

Particularmente, a distância da origem ($x = 0$) para o hiperplano ótimo é dada por

$$\frac{b_0}{||w_0||} \tag{3.9}$$

Se $b_0 > 0$, a origem estará no lado positivo do hiperplano ótimo, caso $b_0 < 0$ a origem estará no lado negativo. Se $b_0 = 0$ o hiperplano passa pela origem. Na Figura 3.7 é dada uma representação geométrica para os resultados algébricos mencionados anteriormente.

A tarefa dada para se obter um hiperplano ótimo é, então, a de encontrar os parâmetros w_0 e b_0 dado um conjunto de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$ [29]. Tendo em mãos os resultados representados geometricamente na figura 3.7, vemos que o par (w_0, b_0) deve satisfazer a seguinte regra:

$$w_0^T x_i + b_0 \geq 0 \quad para \quad d_i = +1 \tag{3.10}$$

$$w_0^T x_i + b_0 < 0 \quad para \quad d_i = -1 \tag{3.11}$$

Como em nosso contexto estamos contando com a restrição de que os padrões são linearmente separáveis, poderemos então alterar os valores de w_o e b_0 nas equações 3.10 e 3.11, além de não realizar nenhuma alteração na equação 3.4.

Os pontos (x_i, y_i) que satisfazem uma igualdade nas equações 3.10 e 3.11 são chamados de vetores de suporte, que dão nome ao método máquina de vetores de suporte. Estes

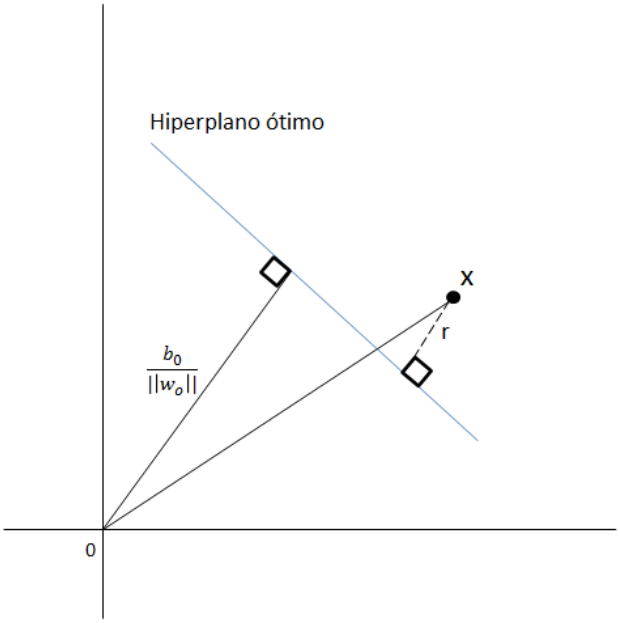


Figura 3.7: representação geométrica das distâncias algébricas de pontos para o hiperplano ótimo em duas dimensões

vetores exercem um papel de destaque na operação desta classe de algoritmos de aprendizagem de máquina [29]. Em termos conceituais, estes vetores de suporte são os pontos mais próximos da superfície de decisão, sendo estes os mais difíceis de serem classificados e possuem uma relação direta na localização ótima da superfície de decisão [29].

Considere um vetor de suporte $x^{(s)}$ com $d^{(s)} = +1$. Logo pela definição temos:

$$g(x^{(s)}) = w_0^T x^{(s)} \pm b_0 = \pm 1 \quad para \quad d^{(s)} = \pm 1 \tag{3.12}$$

Da equação 3.8 a distância algébrica do vetor de suporte $x^{(s)}$ para o hiperplano ótimo é igual a

$$r = \begin{cases} +\frac{1}{||w_0||} & se \quad d^{(s)} = +1 \\ -\frac{1}{||w_0||} & se \quad d^{(s)} = -1 \end{cases} \tag{3.13}$$

Onde o sinal positivo indica que $x^{(s)}$ está no lado positivo do hiperplano ótimo e o sinal negativo indica que o vetor de suporte está no lado negativo. Sendo ρ o valor otimizado da margem de separação entre as duas classes que constituem o conjunto de treinamento τ. Assim, segue da equação 3.13 que:

$$\rho = 2r = \frac{2}{||w_0||} \tag{3.14}$$

equação esta que mostra que maximizar a margem de separação entre as classes é equivalente a minimizar e norma euclidiana do vetor de pesos w [29].

Resumindo, a equação 3.4 que define um hiperplano ótimo é único, no sentido em que o vetor de pesos ótimo w_0 provê a separação máxima possível entre amostras positivas e negativas. Esta condição de otimização é alcançada minimizando a norma euclidiana do vetor de pesos w [29].

3.2.2 Otimização quadrática para localização de hiperplanos ótimos

Nesta subseção será apresentado um método computacionalmente eficiente, proposto por S. Haykin [29] para encontrar hiperplanos ótimos usando o conjunto de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$ sujeito a seguinte condição:

$$d_i(w^T x_i + b) \geq 1 \quad para \quad i = 1, 2, ..., N \tag{3.15}$$

Esta condição combina as equações 3.10 e 3.11 com w ao invés de w_0. Dado esta condição, o problema de otimização a ser resolvido possui o seguinte enunciado:

Dado o conjunto de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$ *encontre os valores ótimos para o vetor de pesos* w *e o viés* b *de forma que estes satisfaçam a seguintes restrições:*

$$d_i(w^T x_i + b) \geq 1 \quad para \quad i = 1, 2, ..., N \tag{3.16}$$

e o vetor de pesos w *minimize a função de custo:*

$$\Phi(w) = \frac{1}{2} w^T w \tag{3.17}$$

Este problema de otimização é chamado de problema primário e é caracterizado da seguinte maneira:

1. A função de custo $\Phi(w)$ é uma função convexa de w;
2. As restrições são lineares em w.

Portanto, é possível resolver este problema usando o método dos multiplicadores de Lagrange [9]. Primeiramente, constrói-se a função lagrangiana:

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i [d_i(w^T x + b) - 1] \tag{3.18}$$

Onde a variável auxiliar positiva α_i é chamada de um multiplicador de Lagrange. A solução para este problema de otimização é determinado pelo ponto de sela da função lagrangiana $J(w, b, \alpha)$ o qual tem de ser minimizado com respeito a w e b e maximizado em relação a α. Assim diferenciando $J(w, b, \alpha)$ com respeito a w e a b e configurando os resultados iguais a zero, alcançamos duas condições de maximização:

$$\text{Condição 1:} \qquad \frac{\partial J(w,b,\alpha)}{\partial w} = 0$$

$$\text{Condição 2:} \qquad \frac{\partial J(w,b,\alpha)}{\partial b} = 0$$

Aplicando a primeira condição de otimização teremos que:

$$w = \sum_{i=1}^{n} \alpha_i d_i x_i \tag{3.19}$$

e aplicando a segunda condição de otimização temos que:

$$\sum_{i=1}^{n} \alpha_i d_i = 0 \tag{3.20}$$

O vetor solução w é definido em termos de uma expansão que envolve as n amostras do conjunto de treinamento. Note, entretanto que embora tenhamos uma solução única em virtude da convexidade da função lagrangiana, o mesmo não pode ser dito sobre o coeficiente de Lagrange α_i.

É importante também notar que no ponto de sela, para cada multiplicador de Lagrange α_i, o produto deste multiplicador com suas restrições desaparece, como mostrado na equação

$$\alpha_i[d_i(w^T x + b) - 1] = 0 \tag{3.21}$$

Com isso, apenas multiplicadores de Lagrange que se encaixam na equação 3.21 podem assumir valores diferentes de zero [29].

Dado um problema de otimização em restrições é possível construir outro problema chamado problema secundário. Este problema secundário tem o mesmo valor ótimo que o problema primário, mas com multiplicadores fornecendo a solução ótima. Segundo o teorema da dualidade [9]:

1. Se o problema primário possui uma solução ótima, o problema secundário também terá uma solução ótima, e os valores ótimos correspondentes serão iguais;
2. Para w_0 ser uma solução primária ótima e α_0 ser uma solução secundária ótima, é necessário e suficiente que w_0 seja possível de ser usado no problema primário e que

$$\Phi(w_0) = J(w_0, b_0, \alpha_0) = \min_{w} J(w_0, b_0, \alpha_0) \tag{3.22}$$

Para postular o problema secundário para o nosso problema primário, primeiramente expandimos a equação 3.18

$$J(w, b, \alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{n} \alpha_i d_i w^T x_i - b \sum_{i=1}^{n} \alpha_i d_i + \sum_{i=1}^{n} \alpha_i \tag{3.23}$$

O terceiro termo do lado direito da equação 3.23 é igual a zero devido a condição da equação 3.20. Além disso, pela equação 3.19 temos que

$$w^T w = \sum_{i=1}^{n} \alpha_i d_i w^T x_i = \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{3.24}$$

Portanto, definindo a função objetivo $J(w, b, \alpha) = Q(\alpha)$, podemos reformular a equação 3.23 como

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{3.25}$$

onde os $\alpha_i's$ são positivos.

Com isso é possível enunciar o problema secundário:

Dado uma amostra do conjunto de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$, *encontre os multiplicadores de Lagrange* $\alpha_i's$ *que maximizem a função objetivo*

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{3.26}$$

Sujeitas às seguintes restrições:

1. $\sum_{i=1}^{n} \alpha_i d_i = 0$
2. $\alpha_i \geq 0$ para $i = 1, 2, ..., n$

Note que o problema secundário é moldado inteiramente em termos dos dados de treinamento e, além disso, a função $Q(\alpha)$ a ser maximizada depende apenas dos padrões de entrada na forma de um conjunto de produtos de pontos $(x_i^T x_j)$, para $i, j = 1, ..., n$.

tendo determinado os multiplicadores de Lagrange ótimos, denotados por $\alpha_{0,i}$, é possível encontrar o vetor de pesos ótimo w_0 usando a equação 3.19, tendo então

$$w_0 = \sum_{i=1}^{n} \alpha_{0,i} d_i x_i \tag{3.27}$$

Para computar o viés ótimo b_0 utilizaremos o w_0 obtido na equação 3.12

$$b_0 = 1 - w_0^T x^{(s)} \quad para \quad d^{(s)} = 1 \tag{3.28}$$

3.2.3 Hiperplano ótimo para padrões não separáveis

Nas subseções anteriores foi discutido a construção de hiperplanos para padrões linearmente separáveis. Nesta subseção será abordado o caso em que dado um conjunto de treinamento, não é possível construir um hiperplano sem erros de classificação. Entretanto é possível encontrar um hiperplano que minimize a probabilidade de haver um erro de classificação [29].

A margem de separação entre classes é dita ser flexível se em um ponto (x_i, d_i) violar a seguinte condição:

$$d_i(w^T x_i + b) \geq 1 \quad para \quad i = 1, 2, ..., N \tag{3.29}$$

Esta violação pode ser de dois tipos:

- Um ponto (x_i, d_i) cai dentro da região de separação mas no lado correto da superfície de decisão (figura 3.8a);
- Um ponto (x_i, d_i) cai no lado incorreto da superfície de decisão (figura 3.8b).

No caso da primeira violação não ocorre um erro de classificação, ao contrário da segunda violação. Para o tratamento destes pontos que violam a equação 3.29, é introduzido um conjunto de variáveis escalares positivas,$\{\xi_i\}_{i=1}^n$, chamadas de variáveis de folga, na definição de hiperplano de separação

$$d_i(w^T x_i + b) \geq 1 - \xi_i \quad para \quad i = 1, 2, ..., N \tag{3.30}$$

As variáveis de folga medem a divergência do ponto para a condição ideal de separação de padrões. Para $0 \leq \xi_i \leq 1$, o ponto cai dentro da região de separação no lado certo da superfície. Para o caso em que $\xi_i > 1$ o ponto cai no lado errado do hiperplano de separação. Caso todas as variáveis tenham $\xi = 0$ o problema se reduz para o caso de padrões linearmente separáveis.

Para minimizarmos o erro de classificação devemos minimizar o custo funcional $\Phi(\xi)$ com respeito ao vetor de pesos w onde

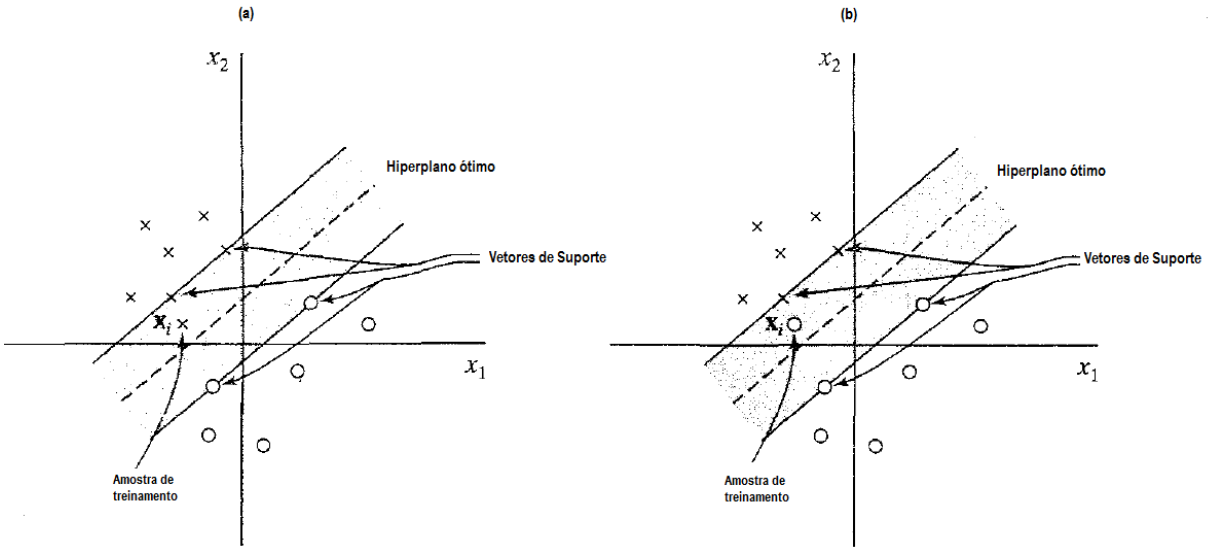


Figura 3.8: (a) O ponto x_i pertencente a classe 1 cai dentro da região de separação, mas no lado correto da superfície de decisão. (b) o ponto x_i pertencente a classe 2 cai do lado incorreto da superfície de decisão.

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \tag{3.31}$$

Onde o parâmetro C controla um *tradeoff* entre complexidade da máquina e o número de pontos não separáveis, onde pode ser visto como uma parâmetro de regularização [29]. Este parâmetro normalmente é escolhido normalmente de forma experimental ou analítica [29].

O problema primário para o caso de padrões não linearmente separáveis pode ser formalmente enunciado da seguinte forma:

Dado uma amostra de treinamento $\{(x_i, d_i)\}_{i=1}^{n}$ *encontre os valores ótimos para o vetor de pesos* w *e o viés* b *de forma que estes satisfaçam a seguinte restrição*

$$d_i(w^T x_i + b) \geq 1 - \xi_i \quad para \quad i = 1, 2, ..., N \tag{3.32}$$

$$\xi_i \geq 0 \quad para \quad todo \quad i \tag{3.33}$$

e que o vetor de pesos w *e as variáveis de folga* ξ_i *minimizem o custo funcional*

$$\Phi(w, \xi) = \frac{1}{2} w^T w + C \sum_{i=1}^{n} \xi_i \tag{3.34}$$

onde C é um parâmetro positivo especificado pelo usuário [29].

Utilizando o método dos multiplicadores de Lagrange e procedendo de forma similar ao caso de padrões linearmente separáveis, pode-se formular o problema secundário para padrões não separáveis como:

Dado uma amostra de treinamento $\{(x_i, d_i)\}_{i=1}^{n}$ *encontre os multiplicadores de lagrange* $\{\alpha_i\}_{i=1}^{n}$ *que maximizem a função objetivo*

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j x_i^T x_j \tag{3.35}$$

sujeita as seguintes restrições

1. $\sum_{i=1}^{n} \alpha_i d_i = 0$
2. $0 \leq \alpha_i \leq C \quad para \quad i = 1, 2, ..., N$

Onde C é um parâmetro positivo especificado pelo usuário.

Note que a única diferença entre o problema secundário de padrões linearmente separáveis dos não linearmente separáveis é a restrição $\alpha_i \geq 0$ é substituída para uma restrição mais rigorosa $0 \leq \alpha_i \leq C$. Sendo assim as computações utilizadas para encontrar os valores ótimos do vetor de pesos w e do viés b funcionam da mesma forma.

A solução ótima para o vetor de pesos w é dada por:

$$w_0 = \sum_{i=1}^{N_s} \alpha_{0,i} d_i x_i \tag{3.36}$$

onde N_s é o número de vetores de suporte. A determinação do valor ótimo para o viés b também segue um procedimento similar ao descrito anteriormente para o caso de padrões linearmente separáveis. A equação 3.21 é reescrita na forma

$$\alpha_i [d_i (w^T x + b) - 1 + \xi_i] = 0 \quad para \quad i = 1, 2, ..., n \tag{3.37}$$

e

$$\mu_i \xi_i = 0 \quad para \quad i = 1, 2, ..., n \tag{3.38}$$

onde μ_i são multiplicadores de Lagrange que foram introduzidos para forçar que as variáveis de folga ξ_i assumissem valores positivos para todo i. No ponto de sela a derivada da função Lagrangiana para o problema primário com respeito a variável ξ_i é igual a zero. Com isso temos que

$$\alpha_i + \mu_i = C \tag{3.39}$$

Combinando as equações 3.38 e 3.39 vemos que

$$\xi_i = 0 \quad se \quad \alpha_i < C \tag{3.40}$$

É possível determinar o viés ótimo b_0 pegando qualquer ponto (x_i, d_i) do conjunto de treinamento para o qual tem-se $0 < \alpha_{0,i} < C$ e, sendo assim $\xi_i = 0$, e usando este ponto na equação 3.37. Entretanto do ponto de vista numérico, é recomendável pegar o valor médio de b_0 resultante de todos pontos do conjunto de treinamento [29].

3.2.4 Construção de uma SVM para reconhecimento de padrões

Segundo S. Haykin [29] A ideia principal do uso de uma SVM para o reconhecimento de padrões se articula em duas operações matemáticas:

1. Mapeamento não linear de um vetor de entrada dentro de um espaço de características de alta dimensão;
2. Construção de um hiperplano para separar as características descobertas na primeira operação.

A primeira operação é realizada de acordo com o teorema de Cover [18], que nos diz que a partir de um espaço de entrada feito de padrões não linearmente separáveis é possível ser transformado em um novo espaço de características onde os padrões são linearmente separáveis com alta probabilidade, se forem satisfeitas as seguintes condições: a transformação não é linear e a dimensão do espaço de característica é alta o suficiente [29]. Estas condições estão embutidas na primeira operação. Já a segunda operação explora a ideia de construir um hiperplano ótimo de acordo com a teoria apresentada na subseção anterior com uma única diferença: O hiperplano separador é agora definido como uma função linear de vetores traçados no espaço de características ao invés do espaço de entrada original. A construção deste hiperplano dependerá da avaliação de um produto interno *kernel*.

Produto interno *kernel*

Seja x um vetor desenhado no espaço de entrada tendo dimensão m_0. Seja $\{\varphi_j(x)\}_{j=1}^{m_1}$ o conjunto de transformações não lineares do espaço de entrada para o espaço de características onde m_1 é a dimensão do espaço de características. Assume-se que $\varphi_j(x)$ é definido a priori para todo j. Dado este conjunto de transformações, pode ser definido um hiperplano agindo como superfície de decisão da seguinte forma:

$$\sum_{j=1}^{m_1} w_j \varphi_j(x) + b = 0 \tag{3.41}$$

Onde $\{w_j\}_{j=1}^{m_1}$ é o conjunto de pesos lineares conectando o espaço de características ao espaço de saída e b é o viés. Podemos simplificar a equação 3.41 como:

$$\sum_{j=0}^{m_1} w_j \varphi_j(x) = 0 \tag{3.42}$$

onde é assumido que $\varphi_0(x) = 1$ para todo x e w_0 é o viés b. A equação 3.42 define a superfície de decisão computada dentro do espaço de características em termos dos pesos lineares da máquina. A quantidade $\varphi_j(x)$ representa a entrada fornecida ao peso w_j via o espaço de características. Defina o vetor $\varphi(x)$ como

$$\varphi(x) = [\varphi_0(x), \varphi_1(x), ..., \varphi_{m_1}(x)]^T \tag{3.43}$$

sendo que para todo x temos $\varphi_0(x) = 1$.

Com efeito, o vetor $\varphi(x)$ representa a imagem induzida do vetor de entrada x no espaço de características como ilustrado na figura 3.9. Assim em termos desta imagem podemos definir a superfície de decisão na sua forma compacta:

$$w^T \varphi(x) = 0 \tag{3.44}$$

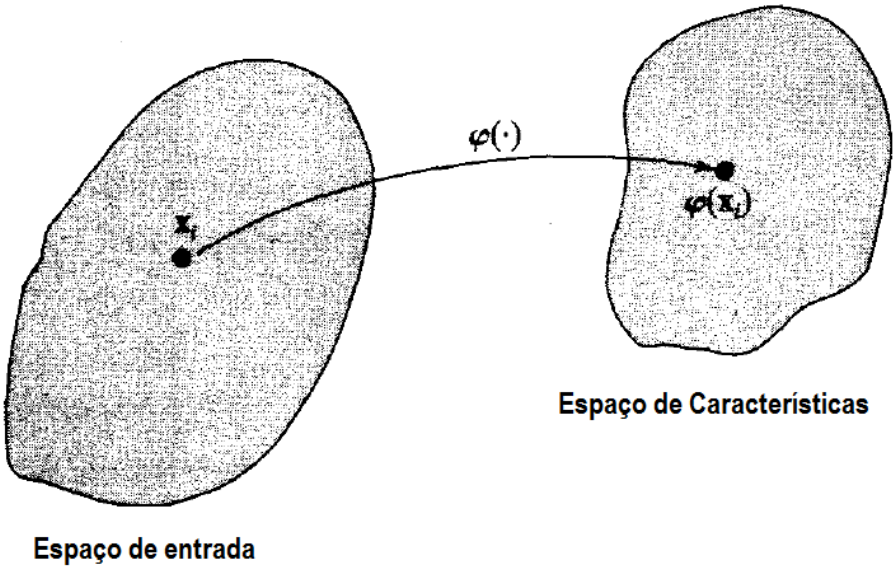


Figura 3.9: Mapeamento não linear $\varphi(.)$ do espaço de entrada para o espaço de características [29]

Alterando a equação 3.19 para a situação envolvendo o espaço de características onde agora procuramos uma forma de separar as características de forma linear, temos:

$$w = \sum_{i=1}^{n} \alpha_i d_i \varphi(x_i) \tag{3.45}$$

Onde o vetor de características $\varphi(x_i)$ corresponde ao padrão de entrada x_i na i-ésima amostra do conjunto de treinamento. Sendo assim, substituindo a equação 3.44 na 3.45, pode-se definir a superfície de decisão computada no espaço de características como:

$$\sum_{i=1}^{n} \alpha_i d_i \varphi^T(x_i)\varphi(x) = 0 \tag{3.46}$$

O termo $\varphi^T(x_i)\varphi(x)$ representa o produto interno de dois vetores induzidos no espaço de características do vetor de entrada x e do padrão de entrada x_i pertencendo a i-ésima amostra do conjunto de treinamento. Com isso temos o produto interno *kernel*, denotado por $K(x, x_i)$, definido como:

$$K(x, x_i) = \varphi^T(x)\varphi(x_i) = \sum_{j=0}^{m_1} \varphi_j(x)\varphi_j(x_i) \quad para \quad i = 1, 2, ..., n \tag{3.47}$$

Desta definição pode-se perceber que o produto interno *kernel* é uma função simétrica isto é $K(x, x_i) = K(x_i, x)$ para todo i. Outro fato importante é que a expansão realizada na equação 3.47 é um caso especial do teorema de Mercer que nos diz se o candidato *kernel* pode ou não ser o produto interno *kernel* em algum espaço e sendo assim admissível para o uso em SVM [29]. O produto interno *kernel* é usado para a construção do hiperplano ótimo no espaço de características sem a necessidade de considerar o espaço de características em sua forma explícita [29].Com isso Podemos reescrever a equação 3.46 na forma:

$$\sum_{i=1}^{n} \alpha_i d_i K(x, x_i) = 0 \tag{3.48}$$

A tabela 3.1 mostra os três mais comuns tipos de SVM com o produto interno *kernel* utilizado em cada uma.

Desenvolvimento de uma máquina de vetores de suporte otimizada

A partir da inserção de um produto interno *kernel* $k(x, x_i)$ na equação 3.48 é possível construir uma superfície de decisão que é não linear no espaço de entrada, mas linear em sua imagem no espaço de características [29]. Com esta inserção é possível agora enunciar a forma secundária para a otimização de uma SVM:

Dado uma amostra de treinamento $\tau = \{(x_1, d_1), ..., (x_i, d_i), ...(x_n, d_n)\}$ *para* $i = 1, 2, ..., n$, *encontre os multiplicadores de Lagrange* $\{\alpha_i\}_{i=1}^{N}$ *que maximize a função objetivo*

Tabela 3.1: Produto interno *kernel* para três tipos de SVMs [29]

Tipo de SVM	Produto interno *kernel* $K(x, x_i)$ para $i = 1, 2, ..., n$	Comentários
Aprendizagem de máquina polinomial	$(x^T x_i + 1)^p$	A potência p é especificada a priori pelo usuário
Rede de função de base radial	$\exp(-\frac{1}{2\sigma^2}\|\|x - x_i\|\|^2)$	A largura σ^2, comum a todos os *kernels* é especificado a priori pelo usuário
Perceptron de duas camadas	$tanh(\beta_0 x^T x_i + \beta_1)$	O teorema de Mercer é satisfeito apenas para alguns valores de β_0 e β_1

$$Q(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j d_i d_j K(x_i, x_j) \tag{3.49}$$

sujeita as seguintes restrições:

1. $\sum_{i=1}^{n} \alpha_i d_i = 0$
2. $0 \leq \alpha \leq C$ para $i = 1, 2, ..., n$

Onde C é um parâmetro positivo especificado pelo usuário.

Encontrando os valores ótimos dos multiplicadores de Lagrange, denotados por $\alpha_{0,i}$, é possível determinar o valor ótimo correspondente ao vetor de pesos linear, w_0, conectando o espaço de características ao espaço de saída adaptando a equação 3.27 para este novo contexto. Para isso, é possível definir o vetor de pesos ótimo w_0 como:

$$w_0 = \sum_{i=1}^{n} \alpha_{0,i} d_i \varphi(x_i) \tag{3.50}$$

Onde $\varphi(x_i)$ é a imagem induzida no espaço de características de x_i. Note que o primeiro componente de w_0 representa o viés ótimo b_0.

3.3 Validação cruzada

Nesta seção descrevemos algumas técnicas de validação cruzada, utilizadas no treinamento de algoritmos de aprendizagem de máquina, notando-se que neste trabalho foi utilizada a validação cruzada k-fold.

A validação cruzada, ou *cross-validation* é um método estatístico para avaliar e comparar algoritmos de aprendizagem dividindo dados em dois segmentos: um segmento é

usado para aprender ou treinar um modelo e o outro é usado para validar o modelo. Tipicamente na validação cruzada os conjuntos de treinamento e validação são rearranjados em sucessivas rodadas de tal forma que cada amostra dos dados tenha a chance de ser validada [62].

Segundo Refaeilzadeh [62], há dois objetivos na validação cruzada:

- Estimar a performance de um modelo de aprendizagem obtido pelos dados disponíveis usando algum algoritmo, isto é, medir generalizabilidade de um algoritmo;
- Comparar a performance de dois ou mais algoritmos diferentes e encontrar o melhor algoritmo a ser usado nos dados disponíveis, ou então comparar a performance de duas ou mais variantes de um modelo parametrizado.

De acordo com estes dois objetivos, várias técnicas foram propostas. Nas subseções a seguir serão descritas algumas destas técnicas.

3.3.1 Re-substituição

Na validação por re-substituição, o modelo de aprendizagem é alcançado utilizando todos os dados disponíveis como conjunto de treinamento e como conjunto de teste. Este processo de validação por utilizar todos os dados em ambas as fases tem grandes chances de causar um (*overfitting*), isto é, a rede perde a capacidade de generalização e, portanto, a capacidade de classificar corretamente dados que não foram apresentados no treinamento [62].

3.3.2 *Holdout validation*

Afim de evitar *overfitting*, um conjunto independente de teste é ideal. Uma abordagem natural é a de dividir os dados disponíveis em duas partes distintas: um para o treinamento e a outra para a fase de testes. Vale ressaltar que os dados para teste não são utilizados durante a fase treinamento. A validação *holdout* evita sobreposição entre os conjuntos de treinamento e de teste permitindo um estimador mais preciso para a performance de generalização de um algoritmo [62]. Entretanto o sucesso deste procedimento está diretamente dependente da escolha dos conjuntos de treinamento e de testes, além do fato de que o modelo gerado na fase de treinamento não "aprende"a partir de todos os dados. Com isto a tarefa de dividir os dados em conjuntos de treinamento e testes pode ser uma tarefa bastante árdua.

3.3.3 *K-Fold cross-validation*

No *k-fold cross-validation* os dados são primeiramente particionados em k segmentos (*folds*) de mesmo tamanho. Após isso k iterações de treinamento e validação são realizadas de tal modo que para cada iteração um segmento dos dados é usado como validação enquanto os $k-1$ segmentos restantes são utilizados como treinamento. Os dados são normalmente estratificados para serem divididos em k segmentos. Estratificação é o processo de rearranjar os dados de forma a assegurar que cada segmento é um bom representante de todos os dados [62].

3.3.4 *Leave-one-out cross validation*

Leave-One-Out Cross Validation (LOOCV) é um caso especial do *k-fold cross-validation* onde k é igual ao número de instâncias dos dados. Em outras palavras em cada iteração quase todos os dados, exceto uma única instância é usada como treinamento e o modelo é testado para esta única instância. O estimador de acurácia obtido pelo LOOCV é conhecido por ser quase não viesado, mas possui alta variância, deixando o estimador pouco confiável. Esta técnica ainda é amplamente utilizada quando os dados disponíveis são muito raros, especialmente na Bioinformática onde há casos onde pouquíssimas amostras estão disponíveis [62].

3.3.5 *Repeated k-fold cross-validation*

Para obter um estimador de performance ou de comparação confiável, um grande número de estimadores é sempre preferido. No *k-fold cross-validation* apenas k estimadores são obtidos. Um método utilizado para aumentar o número de estimadores é executar o *k-fold cross-validation* múltiplas vezes. O dado é embaralhado e re-estratificado antes de cada rodada [62].

Capítulo 4

Projeto para identificar *snoRNAs*

Neste capítulo, serão apresentados os métodos utilizados para a identificação de *snoRNAs* H/ACA box na seção 4.1 e de *snoRNAs* C/D box na seção 4.2. Além disso, será apresentado o método que identifica *snoRNAs* no fungo *Paracoccidioides brasiliensis*. A extração de vetores de características foi realizada usando o snoReport v2.0, sendo tais vetores utilizados para o treinamento de um classificador SVM com a biblioteca libSVM v3.17 [13].

4.1 Identificação de *snoRNAs* H/ACA box

Nesta seção, detalharemos o método utilizado para a identificação do *snoRNAs* H/ACA box.

4.1.1 Extração do vetor de características

A extração do vetor de características para o classificador de *snoRNAs* H/ACA box foi realizada usando a ferramenta snoReport v2.0. Nesta versão foi utilizada o pacote Vienna RNA v2.1.6, em particular o RNAfold [31], responsável pela predição da estrutura secundária dos candidatos a *snoRNAs* H/ACA box.

As condições utilizadas para a predição da estrutura secundária sofreram algumas alterações em relação a versão anterior, onde apenas os boxes H e ACA deveriam permanecer não pareados. Nesta nova versão, a 14ª base a partir da extremidade 5' após os boxes H e ACA não devem ser pareadas, possibilitando a abertura da região de *pocket*, O que permite que o rRNA interaja com essa região alvo do *snoRNA* H/ACA box.

Outras mudanças no snoReport v2.0 em relação as versões anteriores foram: os atributos extraídos para a formação do vetor de características do *snoRNA* H/ACA box, mostrados na tabela 4.1.

Nessa tabela o valor $mfeC$ mostra quanto "esforço" é necessário para forçar uma sequência a se dobrar em uma estrutura secundária de H/ACA box snoRNA. Já os conteúdos AC, GC e AU são usados para separar *ncRNAs* de outros tipos de RNAs. O valor de *zscore*, obtido através do programa RNAz [28], representa a estabilidade termodinâmica da estrutura secundária de *ncRNAs*. Este valor é o desvio padrão da MFE, medida em relação à média de MFE de um conjunto de sequências randomizadas com o mesmo tamanho e composição de bases. Os valores *LseqSize*, *RseqSize*, *LloopSC*, *RloopSC*,

Tabela 4.1: Atributos extraídos de candidatos a *snoRNA* H/ACA box utilizando snoReport v2.0.

$mfeC$	Energia livre mínima (MFE) da estrutura secundária do candidato com restrição
AC	Conteúdo AC
GC	Conteúdo GC
GU	Conteúdo GU
$zscore$	zscore obtido do RNAz
$Hscore$	Escore do box H calculado a partir da matriz de pesos de posição específica
$ACAscore$	Escore do box ACA calculado a partir da matriz de pesos de posição específica
$LseqSize$	Quantidade de nucleotídeos antes do box H
$RseqSize$	Quantidade de nucleotídeos depois do box H e antes do box ACA
$LloopSC$	Tamanho do *loop*, onde é encontrado o *pocket* contendo a região alvo, mais próximo do box H
$RloopSC$	Tamanho do *loop*, onde é encontrado o *pocket* contendo a região alvo, mais próximo do box ACA
$LloopYC$	Simetria do *loop* encontrado perto do box H
$RloopYC$	Simetria do *loop* encontrado perto do box ACA
$nstems$	número de *stems*

$LloopYC$, $RloopYC$ e $nstems$ visam diferenciar estruturas de *snoRNAs* H/ACA box de estruturas de outros *ncRNAs* com estrutura parecida.

4.1.2 Fase de treinamento e teste

A partir da extração do vetor de características dos candidatos a H/ACA snoRNAs, foram criados:

- o conjunto positivo contendo:
  - 101 sequências de H/ACA *snoRNAs*;
- o conjunto negativo contendo:
  - 159 sequências de *snoRNA* C/D box;
  - 1141 sequências de *miRNAs*;
  - 101 sequências obtidos por um procedimento de "embaralhamento" de dinucleotídeos das sequências do conjunto positivo, utilizando o SQUID [22].

Em seguida foi utilizada uma ferramenta do pacote libSVM v3.17 chamada *subset.py*. Este *script python* seleciona de forma aleatória um número específico de amostras. Para dados de classificação, ele provê uma seleção estratificada dos dados para garantir que

os subconjuntos possuam a mesma distribuição de classes. Usando este *script*, os dados foram divididos em duas partes, 60% para treinamento e 40% para teste, permitindo uma melhor avaliação da capacidade de generalização do classificador e, se houver *overfitting*, esse possa ser identificado com mais facilidade.

Depois de separar os dados em um conjunto de treinamento e outro de teste, os dados foram normalizados de -1 a 1 usando a ferramenta *svm-scale* (libSVM v3.17). Após a normalização, foi realizada uma busca em grade para os parâmetros C e γ utilizando a ferramenta *grid.py*, disponível no pacote libSVM v3.17. O *grid.py* é uma ferramenta de seleção de parâmetros para classificação C-SVM usando *kernel* RBF (*radial basis function*), que usa a técnica de validação cruzada (no caso deste trabalho 10-fold) para estimar a acurácia (ou outro critério de performance) para cada combinação de parâmetros de um determinado problema na fase de treinamento. De Acordo com C. Hsu (2010) [33], um método prático para identificar bons parâmetros é tentar combinações de C e γ exponencialmente crescentes. Logo, foi realizada uma varredura por todas as combinações de C e γ com ambos os parâmetros começando de 2^{-15} a 2^{15} com um deslocamento de 2^1 para cada passo da busca em grade (isto é, $2^{-15}, 2^{-14}, 2^{-13}, ...2^{15}$). Após esta primeira varredura, foi realizada uma busca em grade refinada, ou seja, foi realizada uma busca em grade apenas na melhor região da grade, diminuindo o deslocamento de cada passo da busca para $2^{0.1}$.

Após a estimação dos melhores parâmetros C e γ, foi então realizado o treinamento utilizando a ferramenta *svm-train*, que possui uma SVM do tipo C-SVM com *kernel* RBF e com validação cruzada 10-fold. A fase de treinamento possibilitou gerar um classificador, utilizado na fase de testes com a ferramenta *svm-predict* disponível na libSVM 3.17.

Para a fase de análises dos resultados, foi utilizado o programa *eval.cpp*, também disponível na libSVM [48]. A partir deste programa, é possível utilizar diferentes critérios de performance na fase de busca de parâmetros, treinamento e teste. A tabela 4.2 apresenta os critérios de performance disponíveis a partir do *eval.cpp*, onde TP é a quantidade de verdadeiros positivos (uma amostra positiva é classificada como positiva), TN é a quantidade de verdadeiros negativos (uma amostra negativa é classificada como negativa), FP é a quantidade de falsos positivos (uma amostra negativa é classificada como positiva), FN é a quantidade de falsos negativos (uma amostra positiva é classificada como negativa), P é a quantidade de amostras positivas e N é a quantidade de amostras negativas.

Tabela 4.2: Critérios de performance utilizados para a análise dos resultados.

Acurácia	$(TP + TN)/(P + N)$
Precisão	$TP/(TP + FP)$
Recall ou Sensibilidade	$TP/(TP + FN)$
Especificidade	$TN/(TN + FP)$
F-score	$2 * Precisão * Recall/(Precisão + Recall)$
BAC (Balanced ACcuracy)	$(Sensibilidade + Especificidade)/2$
AUC (Area Under Curve)	Área embaixo da curva ROC

Estes critérios possuem grande importância para analisar a performance de um classificador. Por exemplo, especificidade e sensibilidade mostram o quão bom o classificador

é em detectar verdadeiros positivos e verdadeiros negativos, respectivamente.

4.2 Identificação de *snoRNAs* C/D box

Nesta seção, detalhamos o método usado para identificar *snoRNAs* C/D box.

4.2.1 Reimplementação da predição de estrutura secundária e extração do vetor de características

A versão 2.0 do programa snoReport não possuía ainda codificada a predição de estrutura secundária e extração do vetor de características para *snoRNAs* C/D box, apenas para *snoRNAs* H/ACA box. Além disso a versão antiga do snoReport versão 1.2.3 continha alguns problemas de implementação. Foi verificado que na etapa da busca dos boxes C, D', C' e D' eram gerados todas as possibilidades de combinação entre estes boxes em uma sequência e não apenas a melhor combinação. Além disso, eram geradas vetores de características para todas estas possibilidades e não apenas para a melhor, o que prejudicaria a generalização dos dados pelo classificador SVM.

Por isso realizamos uma reimplementação (algoritmo 1) da busca dos boxes C, D', C' e D, na qual para cada sequência, buscamos apenas a melhor combinação dos boxes.

Para a busca dos melhores candidatos a boxes C, D', C' e D de uma sequência, primeiramente foram geradas matrizes de pesos de posições específicas (PWMs) para cada box utilizando um script que varre um arquivo *fasta* contendo sequências de boxes de vários organismos diferentes, onde, para cada nucleotídeo, é calculada a probabilidade de que esse encontre-se em determinada posição do box.

Tendo as PWMs de cada box, é realizada uma varredura na sequência em busca do box C. Para cada conjunto de 7 nucleotídeos da sequência, desde que não sejam os 3 primeiros ou os 3 últimos nucleotídeos, o algoritmo calcula um escore baseado na PWM correspondente. No final da varredura, a sequência de 7 nucleotídeos com a maior pontuação é atribuída como box C. O mesmo procedimento é realizado para os boxes D', C' e D (respeitando o tamanho de cada box), sendo que a busca pelo melhor box D' começa após o box C predito, a busca pelo melhor C' começa após o box D' predito e, por fim, a busca pelo melhor D começa após o box C' e vai até 3 nucleotídeos antes do fim da sequência. Caso o box D não seja encontrado, quando o box D' ou C' preditos estão próximos do fim da sequência, os boxes preditos são descartados e o box D é calculado. Após o box D ser encontrado é feita uma busca entre os boxes C e D para os boxes D' e C'. Caso o box C' não seja encontrado, é buscado primeiramente o box C' e depois o box D'. Caso o box C esteja impossibilitando a busca dos outros boxes por estar próximo ao fim da sequência o box D é encontrado primeiramente e então os outros boxes são encontrados em seguida.

Algoritmo 1: Busca dos boxes C, D', C' e D'.

```
Entrada: Sequência de um candidato a snoRNA C/D box
Saída: boxes C, D', C' e D
início
    melhor_escore = 0;
    para cada sequência de 7nt do candidato, 3nt do inicio até 10nt antes do fim faça
        C_escore =calcular_escore_box_C(sequência de 7 nt);
        se C_escore > melhor_escore então
            box_C = sequência de 7nt
            pos_C = inicio da sequência de 7nt
            melhor_escore = C_escore;
        fim
    fim
    melhor_escore=0;
    para cada sequência de 4nt do candidato, do fim do box C a 7nt antes do fim faça
        D'_escore =calcular_escore_box_D'(sequência de 4 nt);
        se D'_escore > melhor_escore então
            box_D' = sequência de 4nt
            pos_D' = inicio da sequência de 4nt
            melhor_escore = D'_escore;
        fim
    fim
    melhor_escore = 0;
    para cada sequência de 7nt do candidato, do fim do box D' a 10nt antes do fim faça
        C'_escore =calcular_escore_box_C'(sequência de 7 nt);
        se C'_escore > melhor_escore então
            box_C' = sequência de 7nt
            pos_C' = inicio da sequência de 7nt
            melhor_escore = C'_escore;
        fim
    fim
    melhor_score = 0;
    para cada sequência de 4nt do candidato, do fim do box C' a 7nt antes do fim faça
        D_escore =calcular_escore_box_D(sequência de 4 nt);
        se D_escore > melhor_escore então
            box_D = sequência de 4nt
            pos_D = inicio da sequência de 4nt
        fim
    fim
    se o box D não for encontrado então
        buscar o box D, do box C até 10nt antes do final da sequência
        buscar os boxes D' e C' entre o boxes C e D
        se o box C' não for encontrado então
            buscar o box C' entre os boxes C e D
            buscar o box D' entre os boxes C e C'
        fim
    fim
    se o box D' não for encontrado então
        buscar o box D, do inicio da sequência até 10nt antes do final da sequência
        buscar os boxes restantes na seguinte ordem: C, D' e C'
    fim
fim
```

Tendo as posições e sequências de cada box, é realizado um truncamento da sequência. A sequência é truncada 15nt antes do inicio do box C, caso o início desse box esteja mais

que 15nt de distância do começo da sequência, e também é truncada 15 nucleotídeos após o fim do box D, caso o fim desse box esteja mais que 15 nt distante do fim da sequência.

Após a descoberta dos boxes e truncamento da sequência, são realizadas duas predições de estrutura secundária, uma sem nenhuma restrição de pareamento e outra restringindo o pareamento de todos os nucleotídeos que se encontram entre o início do box C e o fim do box D. A predição da estrutura secundária é realizada utilizando-se o programa RNAfold, do pacote Vienna RNA. As sequências com estruturas secundárias restritas com *MFE* igual a zero são descartadas, pois neste caso o RNAfold não consegue montar a estrutura secundária, mantendo a estrutura primária da sequência, o que poderia afetar a fase de treinamento.

Os valores da média e de desvio padrão da *mfeC* foram calculados a partir do programa RNAz [28]. Outros atributos do vetor de características também foram calculados como o conteúdo GC, distância entre os boxes, e tamanho do *stem* terminal da estrutura secundária da sequência, notando-se que todos esses atributos ajudam na diferenciação de um C/D box snoRNA de outros *ncRNAs*. Na tabela 4.3 estão descritos os atributos presentes no vetor de features de cada sequência.

Tabela 4.3: Atributos extraídos de candidatos a *snoRNA* C/D box utilizando snoReport v2.0.

mfe	MFE da estrutura secundária do candidato sem restrição
$mfeC$	MFE da estrutura secundária do candidato com restrição
E_{avg}	Média da MFE
E_{stdv}	Desvio padrão da MFE
ls	Tamanho do *stem* terminal
Dcd	Distância do box C ao box D
Dcd'	Distância do box C ao box D'
$Dd'c'$	Distância do box D' ao box C'
$Dc'd$	Distância do box C' ao box D
C_{score}	Escore do box C
D'_{score}	Escore do box D'
C'_{score}	Escore do box C'
D_{score}	Escore do box D
GC	Conteúdo GC

4.2.2 Fase de treinamento e teste

A partir da extração do vetor de características dos candidatos a C/D box snoRNAs, foram criados:

- o conjunto positivo contendo:
  - 115 sequências de C/D *snoRNA*;
- o conjunto negativo contendo:

– 45 sequências de *snoRNA* H/ACA box;

– 473 sequências de *miRNAs*;

– 60 sequências obtidos por um procedimento de "embaralhamento" de dinucleotídeos das sequências do conjunto positivo, utilizando o programa shuffleseq, disponível no pacote EMBOSS [66].

Os mesmos procedimentos apresentados para a geração de um classificador de *snoRNA* H/ACA box foram utilizados para a geração do classificador de C/D box, isto é, os dados foram separados em dois conjuntos de maneira estratificada: o primeiro com 60% dos dados para treinamento e o segundo com 40% para teste usando o script *subset.py*. Após isto, os dados foram normalizados de -1 a $+1$ pelo programa *svm-scale* e então foi realizada uma busca em grade, utilizando o script *grid.py* com validação cruzada 10-fold. Em seguida foi realizado o treinamento com o programa *svm-train*, utilizando C-SVM com *kernel* RBF e validação cruzada 10-fold. o próximo passo foi realizar a fase de testes utilizando *svm-predict* usando o classificador gerado pelo *svm-train*. Os mesmos critérios de performance utilizados para avaliar o classificador de *snoRNAs* H/ACA box (tabela 4.2) também foram utilizados para o classificador de *snoRNAs* C/D box.

O *workflow* do snoReport reimplementado é apresentado na figura 4.1

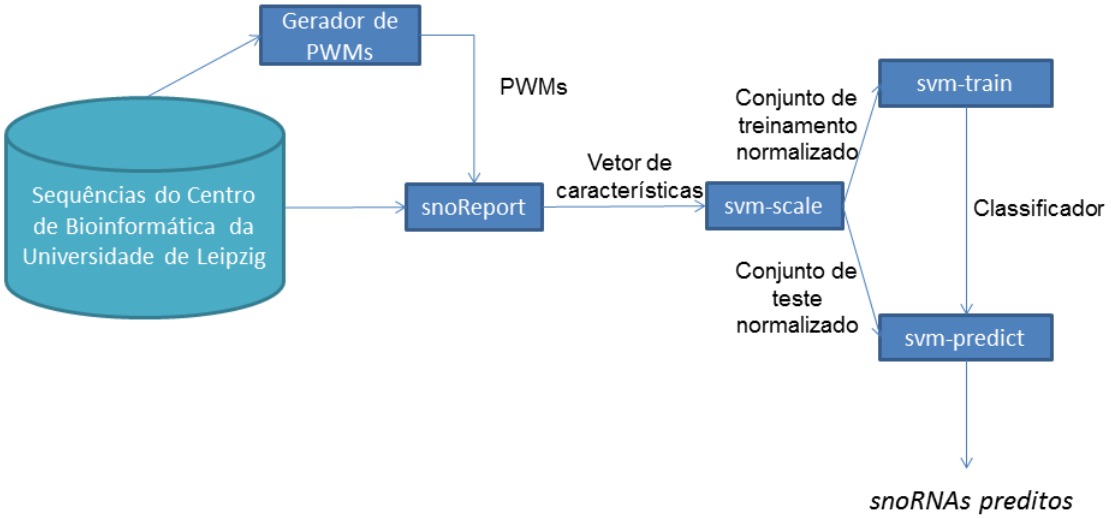


Figura 4.1: *Workflow* usado pelo snoReport reimplementado.

4.3 Um estudo de caso: o fungo *P. brasiliensis*

Foi realizado um estudo de caso com a finalidade de encontrar *snoRNAs* C/D box e *snoRNAs* H/ACA box no fungo *Paracoccidioides brasiliensis*, um fungo dimórfico (com as formas de micélio e levedura) que causa uma micose endêmica nos países tropicais [25]. Esse fungo é estudado no Laboratório de Biologia Molecular do Instituto de Biologia da

UnB, que vem contribuindo significativamente para ampliar o conhecimento sobre este patógeno. Os dados utilizados no experimento realizado neste trabalho foram retirados do banco de dados *Paracoccidioides brasiliensis database* do Instituto BROAD [58].

Como os *snoRNAs* aparecem nas regiões intrônicas dos organismos, foi implementado um programa em C chamado *extrator_de_introns* que, a partir do genoma do fungo e de um arquivo no formato *gtf*[1], produz um arquivo *fasta* contendo todos os introns desse genoma. De forma genérica, esse programa encontra no genoma todas as posições de códon de início, códon de parada e regiões exônicas, e salva as regiões intrônicas, as que estão entre os exons, descartando trechos entre o códon de início e o primeiro exon de uma região codificadora, assim como entre o último exon e o códon de parada.

Após isso, os introns são submetidos ao snoReport reimplementado, que gera os vetores de características para *snoRNAs* C/D box e *snoRNAs* H/ACA box. Em seguida, os vetores de características são normalizados utilizando *svm-scale* com a mesma faixa de valores utilizada no treinamento. Por fim, a ferramenta *svm-predict* prediz *snoRNAs* putativos para ambas as classes de *snoRNAs*.

Lembramos que a predição da estrutura secundária realizada pelo snoReport é feita pelo RNAfold, que recebe como entrada uma sequência e uma restrição usada para direcionar o dobramento em uma estrutura secundária específica. Com o intuito de visualizar a estrutura secundária dos *snoRNAs* putativos encontrados, foi utilizado o snoReport no modo *verbose* com os *snoRNAs* identificados para identificar as restrições de cada sequência. Assim, foi possível gerar uma visualização dos *snoRNAs* putativos identificados.

[1] O arquivo *.gtf* possui informações de regiões codificadoras de proteínas e de exons presentes no genoma de um organismo

Capítulo 5

Resultados e Discussão

Neste capítulo, apresentamos avaliações das SVMs para *snoRNAs* H/ACA box (seção 5.1) e *snoRNA* C/D box (seção 5.2). A seção 5.3 apresenta os *snoRNAs* encontrados no fungo *P. brasiliensis*.

5.1 Identificação de *snoRNA* H/ACA box

Foram realizadas diversas buscas em grade com a finalidade de estimar os melhores parâmetros C e γ utilizando-se diversos critérios de performance. Os critérios foram: acurácia, BAC (acurácia balanceada), F-score e uma combinação de busca em grade de acurácia e uma busca fina com precisão, de forma a reduzir falsos positivos.

As figuras 5.1, 5.2, 5.3 e 5.4 mostram as buscas em grade realizadas utilizando a acurácia como critério de performance. Nesses gráficos, estão mapeadas todas as combinações de C e γ em uma determinada faixa de valores. Por exemplo, na figura 5.1 os valores são de 2^{-15} a 2^{15} com deslocamento de 2^1 para cada passo da busca, onde cada linha no gráfico representa a faixa de valores que alcançam determinada taxa de acurácia no treinamento. A linha verde nos gráficos representa combinações de C e γ que geram classificadores com acurácia de 98%. Após a primeira busca em grade, foi realizada uma busca mais refinada na região que contém 98% de acurácia (figura 5.2) e, a partir desse gráfico, foram geradas mais duas buscas em grades ainda mais refinadas em duas regiões de interesse (figuras 5.3 e 5.4).

A partir da busca em grade, foram gerados 10 classificadores utilizando diferentes critérios de performance para estimar os parâmetros escolhidos. Nas tabelas 5.1 e 5.2 são apresentados os resultados da fase de treinamento e de teste, respectivamente. Além disso, também são apresentados os resultados utilizando os parâmetros $C = 1$ e $\gamma = 2$ usados por Hertel (2008) [30]. Para todos os casos, foram utilizados 60% dos dados para treinamento e 40% dos dados para teste, utilizando validação cruzada 10-fold para o treinamento.

O classificador 1 utiliza os parâmetros C e γ usados no trabalho de Hertel (2008) [30]. Já os classificadores 2, 3, 4 e 5 foram obtidos utilizando os parâmetros encontrados na busca em grade com o critério acurácia. Nos classificadores 6 e 7 os parâmetros foram encontrados na busca em grade utilizando BAC como critério. Os classificadores 8 e 9 foram obtidos utilizando os parâmetros encontrados na busca em grade com critério F-score. Por fim, o classificador 10 foi obtido utilizando os parâmetros encontradas por uma

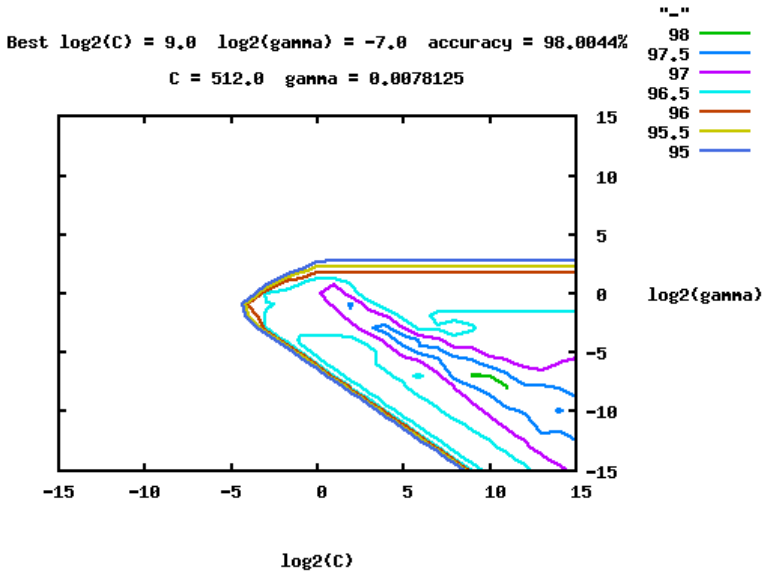


Figura 5.1: Primeira busca em grade realizada para encontrar os melhores parâmetros para o classificador de *snoRNA* H/ACA box.

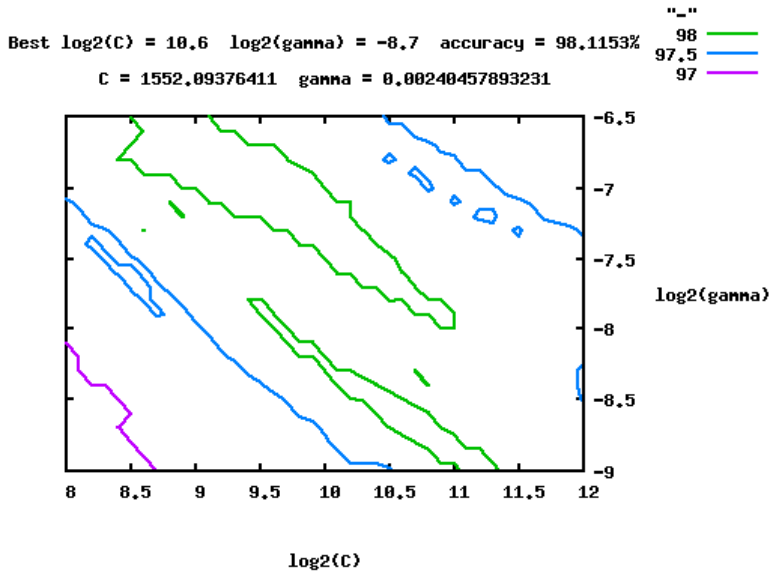


Figura 5.2: Busca em grade mais refinada na região com acurácia de 98% da figura 5.1.

busca em grade com acurácia seguida de uma busca em grade mais refinada com precisão como critério.

Nota-se que praticamente em todos os critérios de performance em ambas as fases de treinamento e teste, os classificadores gerados a partir da busca de parâmetros utilizando busca em grade com validação cruzada 10-fold tiveram melhores resultados, com exceção dos critérios de precisão, especificidade e AUC (na fase de testes). Contudo um classificador com precisão e especificidade de 100% mas um valor mais baixo de sensibilidade parece não estar classificando quase nenhuma amostra do conjunto positivo e negativo na classe positiva. Ao contrário, classifica a maioria das amostras como classe do conjunto

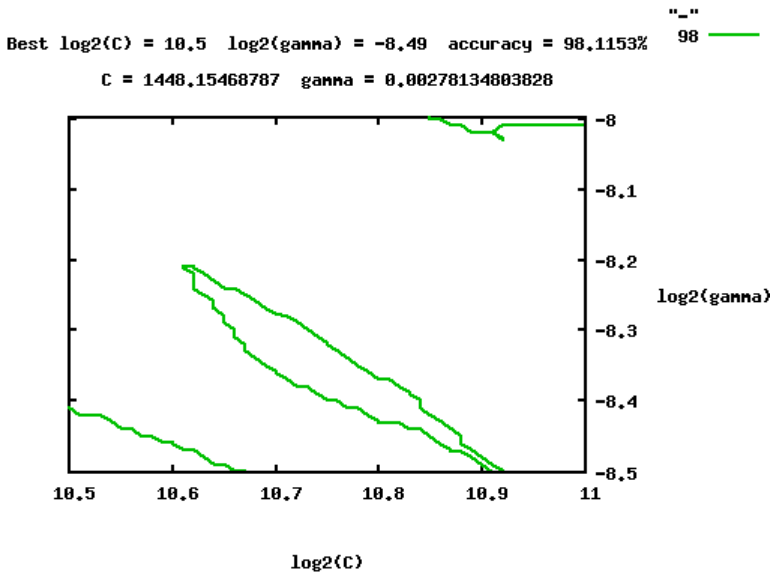


Figura 5.3: Busca em grade mais refinada em uma das regiões com acurácia de 98% da figura 5.2.

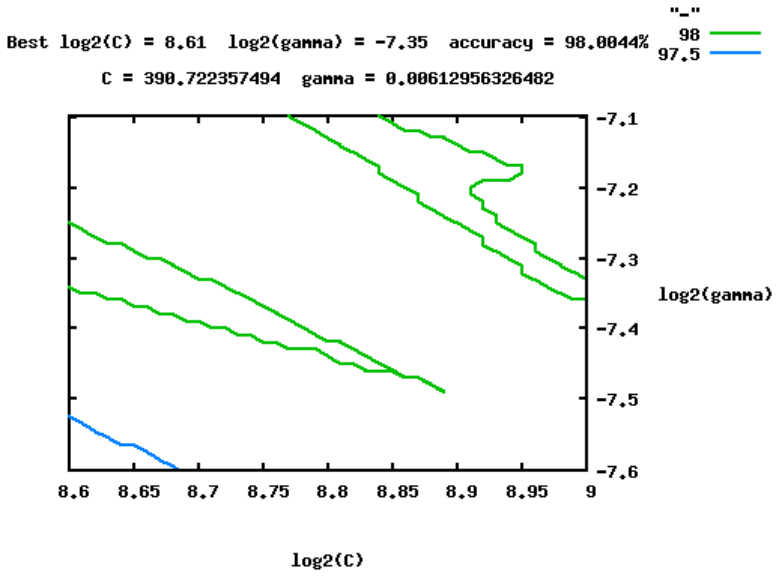


Figura 5.4: Busca em grade mais refinada em uma das regiões com acurácia de 98% da figura 5.2.

negativo, anulando falsos positivos e aumentando significativamente a taxa de falsos negativos. isso explica o alto valor da área de baixo da curva ROC (AUC), pois o eixo referente a especificidade torna-se máximo.

Observando os classificadores, nota-se que possuem resultados bastante próximos. A partir destes resultados, foi escolhido como melhor o classificador 5, obtido através dos parâmetros C e γ encontrados na busca em grade utilizando como critério de performance a acurácia (na figura 5.4), pois possui o maior valor de acurácia, precisão, especificidade e F-score na fase de testes. Embora possua a taxa um pouco menor de sensibilidade, o

Tabela 5.1: Resultados da fase de treinamento para a identificação de *snoRNAs* H/ACA box, observando-se que a primeira coluna indica os classificadores utilizando determinado critério para identificar os parâmetros C e γ, obtendo a acurácia (A), precisão (P), sensibilidade (S), especificidade (E), F-score, BAC e AUC.

	Critério	C	γ	A (%)	P (%)	S (%)	E (%)	F-score%	BAC%	AUC%
1	-	1	2	96.90	94.60	57.38	99.76	71.43	78.57	96.44
2	A	512	0.0078125	98.00	89.09	80.33	99.29	84.48	89.81	99.29
3	A	1552.09376411	0.00240457893231	98.12	89.29	81.97	99.29	85.47	90.63	99.15
4	A	1448.15468787	0.00278134803828	98.12	89.29	81.97	99.29	85.47	90.63	99.21
5	A	390.722357494	0.00612956326482	98.00	90,57	78.69	99.41	84.21	89.05	99.11
6	BAC	16384.0	0.001953125	97.89	85.00	83.61	98.93	84.30	91.27	99.24
7	BAC	11585.237503	0.00240457893231	97.89	85	83.61	98.93	84.30	91.27	99.25
8	F-score	1024	0.0078125	98.00	87.72	81.97	99.17	84.75	90.57	99.32
9	F-score	415.873226934	0.0103086555529	98.12	89.29	81.97	99.29	85.47	90.63	99.32
10	A e P	5792.61875148	0.00296038391897	97.78	83.61	83.61	98.81	83.61	91.21	99.25

Tabela 5.2: Resultados da fase de testes para a identificação de *snoRNAs* H/ACA box, observando-se que a primeira coluna indica os classificadores utilizando determinado critério para identificar os parâmetros C e γ, obtendo a acurácia (A), precisão (P), sensibilidade (S), especificidade (E), F-score, BAC e AUC.

	Critério	A (%)	P (%)	S (%)	E (%)	F-score%	BAC%	AUC%
1	-	97.17	100.00	57.50	100.00	73.02	78.75	98.62
2	A	96.83	75.61	77.50	98.21	76.54	87.86	98.27
3	A	97.17	79.49	77.50	98.57	78.48	88.04	97.59
4	A	97.17	79.49	77.50	98.57	78.48	88.04	97.66
5	A	97.33	83.33	75.00	98.93	78.95	86.96	98.14
6	BAC	96.67	73.81	77.50	98.04	75.61	87.77	97.68
7	BAC	96.83	75.61	77.50	98.21	76.54	87.86	97.76
8	F-score	96.83	75.61	77.50	98.21	76.54	87.86	98.22
9	F-score	96.83	75.61	77.50	98.21	76.54	87.86	98.25
10	A e P	96.67	73.81	77.50	98.04	75.61	87.77	98.05

classificador 5 possui maior valor de precisão e especificidade, suficiente para que o F-score possua o melhor valor entre os classificadores.

5.2 Identificação de *snoRNA* C/D box

Na fase de implementação foram geradas matrizes de posições específicas (PWMs) para cada box existente numa molécula de *snoRNA* C/D box, mostradas nas figuras 5.5 e 5.6.

$$C = \begin{bmatrix} 0.3922 & 0.1569 & 0.2549 & 0.1961 \\ 0.0784 & 0.0980 & 0.0588 & 0.7647 \\ 0.0000 & 0.0392 & 0.9412 & 0.0196 \\ 0.9608 & 0.0196 & 0.0000 & 0.0196 \\ 0.0588 & 0.0784 & 0.1176 & 0.7451 \\ 0.0980 & 0.0000 & 0.8235 & 0.0784 \\ 0.7451 & 0.0588 & 0.1176 & 0.0784 \end{bmatrix} \quad C' = \begin{bmatrix} 0.2380 & 0.2620 & 0.2380 & 0.2268 \\ 0.1134 & 0.0719 & 0.0767 & 0.7061 \\ 0.0799 & 0.1246 & 0.6390 & 0.1246 \\ 0.5479 & 0.1406 & 0.1422 & 0.1358 \\ 0.1422 & 0.1070 & 0.1550 & 0.5623 \\ 0.1709 & 0.0895 & 0.5351 & 0.1709 \\ 0.5112 & 0.1502 & 0.1438 & 0.1597 \end{bmatrix}$$

Figura 5.5: PWMs dos boxes C e C'.

$$D = \begin{bmatrix} 0.1250 & 0.6250 & 0.1250 & 0.1250 \\ 0.1250 & 0.1250 & 0.1250 & 0.6250 \\ 0.0000 & 0.0000 & 1.0000 & 0.0000 \\ 0.8750 & 0.1250 & 0.0000 & 0.0000 \end{bmatrix} \quad D' = \begin{bmatrix} 0.0644 & 0.1416 & 0.0687 & 0.0987 \\ 0.0773 & 0.0515 & 0.0687 & 0.1760 \\ 0.0858 & 0.0601 & 0.1717 & 0.0558 \\ 0.1845 & 0.0601 & 0.0644 & 0.0644 \end{bmatrix}$$

Figura 5.6: PWMs dos boxes D e D'.

Cada linha de cada uma das quatro PWMs representa uma posição do box e cada coluna representa a probabilidade de um nucleotídeo aparecer naquela posição. A primeira coluna refere-se ao nucleotídeo A, a segunda ao nucleotídeo C, a terceira ao nucleotídeo G e a última coluna ao nucleotídeo U. Ao observar as PWMs é possível notar que os boxes C' e D' são menos conservados que os boxes C e D, como observado na literatura [67], pois possuem valores de probabilidade menor do que os encontrados nos boxes C e D.

Nas fases de treinamento e teste, assim como no classificador para *snoRNA* H/ACA box, foram realizadas diversas buscas em grade com a finalidade de estimar os melhores parâmetros C e γ utilizando-se diversos critérios de performance. Os critérios foram: acurácia, BAC, F-score e uma combinação de busca em grade de acurácia e sensibilidade. As figuras 5.7, 5.8 e 5.9 mostram as buscas em grade realizadas utilizando como critério de performance a acurácia. A figura 5.7 representa a primeira busca em grade. Uma busca mais refinada em grade na região de maior acurácia da figura 5.7, gerando o gráfico da figura 5.8. Uma busca ainda mais refinada gerou o gráfico da figura 5.9.

A partir das buscas em grade, foram então gerados 9 classificadores utilizando diferentes critérios de performance para estimar os parâmetros escolhidos. Nas tabelas 5.3 e 5.4 são apresentados os resultados da fase de treinamento e de teste, respectivamente. Além disso também são apresentado os resultados utilizando os parâmetros $C = 1$ e $\gamma = 2$ usados por Hertel (2008) [30]. Para todos os casos, foram utilizados 60% dos dados para treinamento e 40% dos dados para teste, utilizando validação cruzada 10-fold para o treinamento.

O classificador 1 utiliza os parâmetros C e γ usados no trabalho de Hertel (2008) [30]. Já os classificadores 2, 3 e 4 foram obtidos utilizando os parâmetros encontrados na busca em grade com o critério de performance sendo a acurácia. Nos classificadores 5 e 6, os parâmetros foram encontrados utilizando BAC como critério de performance. Para os classificadores 7 e 8, o critério usado para a busca em grade foi o F-score. Por fim, o

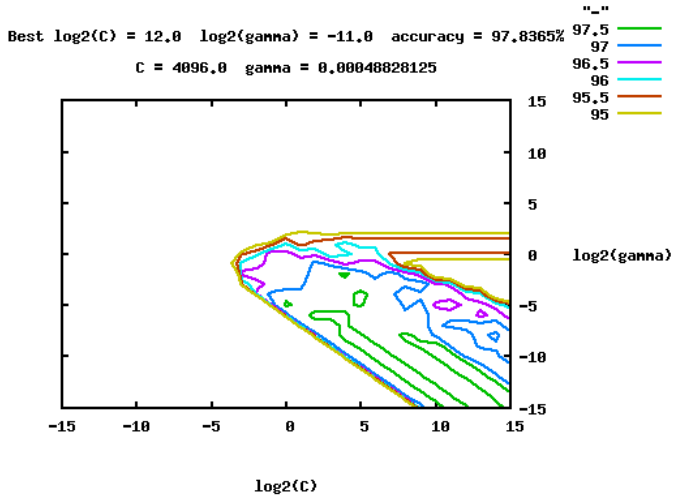


Figura 5.7: A primeira busca em grade realizada para encontrar os melhores parâmetros para o classificador de *snoRNA* C/D box.

Tabela 5.3: Resultados da fase de treinamento para a identificação de *snoRNAs* C/D box, observando-se que a primeira coluna indica os classificadores utilizando determinado critério para identificar os parâmetros C e γ, obtendo a acurácia (A), precisão (P), sensibilidade (S), especificidade (E), F-score, BAC e AUC.

	Critério	C	γ	A (%)	P (%)	S (%)	E (%)	F-score%	BAC%	AUC%
1	-	1	2	96.15	96.55	80.00	99.42	87.50	89.71	98.75
2	A	4096	0.0048828125	97.84	94.20	92.86	98.84	93.53	95.85	97.33
3	A	16384	0.00106268379309	98.08	94.29	94.29	98.84	94.29	96.57	97.34
4	A	20171.0700682	0000863167457503	98.08	94,29	94.29	98.84	94.29	96.57	97.34
5	BAC	4096	0.0048828125	97.84	94.20	92,86	98.84	93.53	95.85	97.33
6	BAC	17559.9364102	0.000991519038521	98.08	94.29	94.29	98.84	94.29	96.57	97.34
7	F-score	4096	0.0048828125	97.84	94.20	92,86	98.84	93.53	95.85	97.33
8	F-score	53231.7730476	0.000327079303753	98.08	94.29	94.29	98.84	94.29	96.57	97.34
9	A e S	4705.06846207	000375213667307	98.08	94.29	94.29	98.84	94.29	96.57	97.33

classificador 9 foi gerado através dos parâmetros encontrados por uma busca em grade com critério de acurácia seguida de uma busca em grade com critério de sensibilidade.

Novamente nota-se que todos os classificadores utilizando a abordagem da busca em grade para a estimação dos parâmetros C e γ obtiveram resultados melhores em praticamente todos os critérios de performance, especialmente no critério de sensibilidade na fase de teste, onde o segundo classificador teve uma melhora de 24.24% em relação ao primeiro classificador. Nota-se também que muitos resultados foram idênticos, não havendo tanta importância ao critério de performance usado na busca em grade.

Os classificadores que possuíram melhor eficácia foram os classificadores 2, 5 e 7. O fato de que na fase de testes a precisão e especificidade alcançarem 100% não indica problema, visto que o valor de sensibilidade alcançou uma alta taxa de acerto (91,111%) alcançando uma área de curva ROC de 99.05% e F-score de 95.56%, para um conjunto

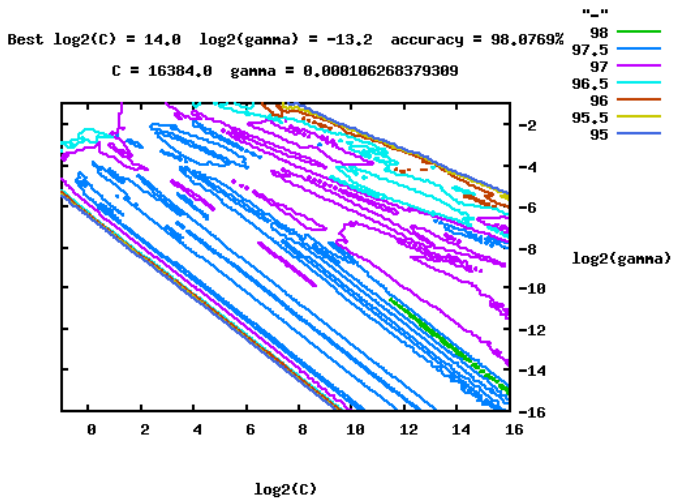


Figura 5.8: A partir da primeira busca em grade (figura 5.7), foi realizada uma busca em grade mais refinada na região com mais acurácia.

Tabela 5.4: Resultados da fase de testes para a identificação de *snoRNAs* C/D boxobservando-se que a primeira coluna indica os classificadores utilizando determinado critério para identificar os parâmetros C e γ, obtendo a acurácia (A), precisão (P), sensibilidade (S), especificidade (E), F-score, BAC e AUC.

	Critério	A (%)	P (%)	S (%)	E (%)	F-score%	BAC%	AUC%
1	-	95.67	100.00	73.33	100.00	84.62	86.67	99.02
2	A	98.56	100.00	91.11	100.00	95.35	95.56	99.05
3	A	98.20	100.00	88.88	100.00	94.12	94.44	99.01
4	A	98.20	100.00	88.89	100.00	94.12	94.44	99.01
5	BAC	98.56	100.00	91.11	100.00	95.35	95.56	99.05
6	BAC	98.20	100.00	88.89	100.00	94.12	94.44	99.00
7	F-score	98.56	100.00	91.11	100.00	95.35	95.56	99.05
8	F-score	98.20	100.00	88.89	100.00	94.12	94.44	98.99
9	A e S	98.20	100.00	88.89	100.00	94.12	94.44	99.03

de dados não utilizado na fase de treinamento.

5.3 Um estudo de caso: o fungo *P. brasiliensis*

O programa *extrator_de_introns* encontrou 30.157 introns no genoma do fungo *Paracoccidioides brasiliensis*, que foram submetidos ao snoReport reimplementado. Algumas características gerais sobre o genoma do fungo *P. brasiliensis* são descritos na tabela 5.5. Foram encontrados 188 *snoRNAs* C/D box putativos e 6.007 *snoRNAs* H/ACA box putativos. Na figura 5.10, mostramos o dobramento 5 *snoRNAs* C/D box putativos.

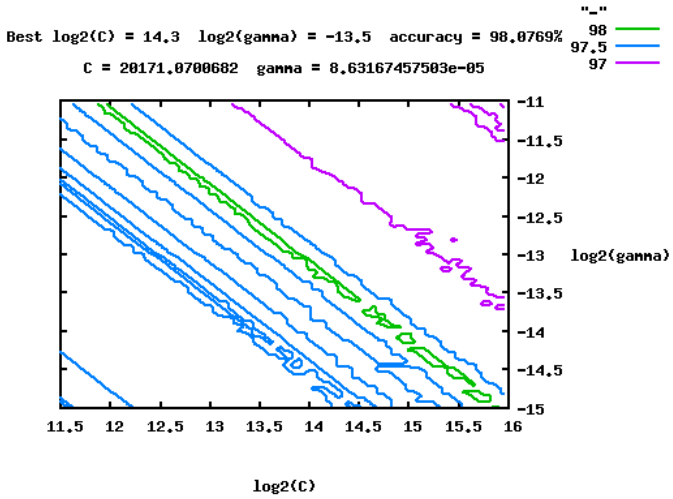
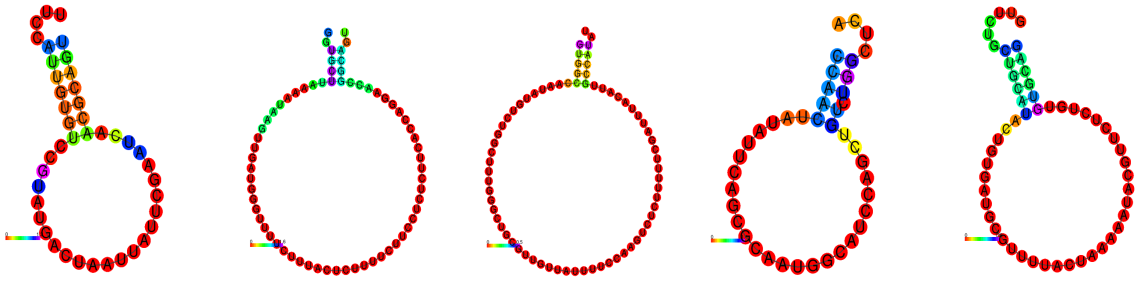


Figura 5.9: A partir da segunda busca em grade (figura 5.8), foi realizada outra busca em grade na região de maior acurácia.

Tabela 5.5: Características gerais sobre o genoma do fungo *P. brasiliensis*.

Tamanho do genoma	31,4 Mb
Quantidade de introns encontrados	30.157
Tamanho médio dos introns	212 pb
Tamanho do maior intron	29.843 pb
Tamanho do menor intron	2 pb
Desvio padrão do tamanho dos introns	713 pb



(a) (b) (c) (d) (e)

Figura 5.10: *snoRNAs* C/D box putativos encontrados no genoma do *Paracoccidioides brasiliensis*, dobrados pelo RNAfold.

Capítulo 6

Conclusão

Neste trabalho, aprimoramos o snoReport usando os dados de *ncRNAs* disponibilizados pelo Centro de Bioinformática da Universidade de Leipzig e novas versões das ferramentas utilizadas no *workflow*. Essas alterações trouxeram melhorias significativas nos classificadores SVM de ambas as classes de *snoRNAs*, (*snoRNA* C/D box e *snoRNA* H/ACA box). Foram utilizados os pacotes Vienna RNA (v2.16) e a biblioteca libSVM (v3.17). Além disso, foram obtidas novas PWMs e foi reimplementado o gerador de vetores de características de *snoRNA* C/D box.

Foi realizado o treinamento da SVM do snoReport com os novos dados e os novos atributos dos vetores de características, o que melhorou significativamente a performance das SVMs, sobretudo com a busca em grade dos parâmetros C e γ da SVM. Foi feito também um estudo de caso com o fungo *Paracoccidioides brasiliensis*, tendo sido identificados novos *snoRNAs* ainda não conhecidos. Esse estudo identificou 188 *snoRNAs* C/D box putativos e 5.929 *snoRNAs* H/ACA box. Nesse último caso, devem ser incluídas restrições da estrutura secundária que não foram ainda implementadas, que geraram uma grande quantidade de falso positivos.

6.1 Contribuições

A nova reimplementação do *snoReport* e uma abordagem mais refinada para a etapa de treinamento permitiram melhorar os critérios de avaliação de performance, tanto do *snoRNAs* C/D box quanto do *snoRNAs* H/ACA box. Além disso, foram identificados *snoRNAs* no fungo *P. brasiliensis*.

6.2 Trabalhos futuros

As perspectivas para este trabalho são:

- Testar diferentes algoritmos de aprendizagem de máquina no *snoReport*, em busca de resultados melhores que a SVM;
- Aumentar o conjunto de dados positivos de ambas as classes de *snoRNA* (*snoRNA* C/D box e *snoRNA* H/ACA box), visto que o conjunto de dados negativos é bem maior;

- Testar a nova reimplementação com os dados utilizados em Hertel [30];
- Buscar novos atributos para o vetor de características de ambas as classes, a fim de melhor diferenciar *snoRNAs* de outras sequências;
- Aprimorar a identificação da região de *pocket* do *snoRNA* H/ACA box, uma vez que essa região pode estar localizada entre 13 a 16 nt após os boxes H e ACA [76], contudo, o snoReport assume que a região está localizada a 14 nt após os boxes H e ACA;
- Aprimorar a identificação de *snoRNAs* C/D box em organismos (exemplo: figura 5.10);
- Entrar em contato com biólogos para realizarem experimentos de bancada de Biologia Molecular para comprovar se os *snoRNAs* preditos são de fato *snoRNAs*.

Referências

[1] Genética Molecular. GBOL. http://www.ufv.br/dbg/gbolhtm/gbol26.htm Acessado em 21/02/2014. vi, 9

[2] Human Brain Project. disponível em: https://www.humanbrainproject.eu/pt/home, 2013. 26

[3] M. C. Accardo et al. A computational search for box C/D snoRNA genes in the *drosophila melanogaster* genome. *Bioinformatics*, 20(18):3293–3301, 2004. 22

[4] S. F. Altschul et al. Basic Local Alignment Search Tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. 17

[5] P. A. Alvarez. *Pipelines para transcritomas obtidos por sequenciadores de alto desempenho*. Monografia de Graduação. Departamento de Ciência da Computação. Universidade de Brasília, 2009. vi, 1, 6

[6] W. C. Arruda. *ncRNA-Agents: Anotação de RNAs não codificadores baseado em Sistemas Multiagente*. Exame de qualificação para o curso de Doutorado em Informática. Departamento de Ciência da Computação. Universidade de Brasília, 2012. ix, 10, 11, 12, 14

[7] T. Bailey and C. Elkan. Fitting a mixture model by expectation maximization to discover motifs in biopolymers. In *Proceedings of the Second International Conference on Intelligent Systems for Molecular Biology*, pages 28–36. AAAI Press, 1994. 17

[8] D. A. Benson et al. Genbank. *Nucleic acids research*, 37(Database issue):D26–31, 2009. 12

[9] D. P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods (Optimization and Neural Computation Series)*. Athena Scientific, 1 edition, 1996. 37, 38

[10] Só Biologia. Células nervosas. http://www.sobiologia.com.br/conteudos/Fisiologia Animal/nervoso2.php. Acessado em 07/01/2014. vii, 26

[11] S. W Burge et al. Rfam 11.0: 10 years of RNA families. *Nucleic Acids Research*, 41(Database-Issue):226–232, 2013. 12, 17

[12] M. Carlos et al. Small nucleolar RNAs U32a, U33, and U35a are critical mediators of metabolic stress. *Cell Metab*, 14(1):33–44, 2011. 16

[13] C. C. Chang and C. J. Lin. LIBSVM: a library for Support Vector Machines, 2001. Software disponível em: http://www.csie.ntu.edu.tw/ cjlin/libsvm/ Acessado em 07/01/2014. 3, 17, 49

[14] P. Clote and R. Backofen. *Computational Molecular Biology: An Introduction*. John Willey & sons Ltd, 2000. 7, 8

[15] J. R. Cole. et al. The ribosomal database project (rdp-ii): previewing a new autoaligner that allows regular updates and the new prokaryotic taxonomy. *Nucleic Acids Research*, 31(1):442–443, 2003. 11

[16] G. M. Cooper. *The Cell - A Molecular Approach 2nd Edition*. Sunderland (MA): Sinauer Associates, 2000. 15

[17] R. Cotter. Bacterial genetics and growth. http://www.pc.maricopa.edu/Biology/rcotter/BIO%20205/LessonBuilders/Chapter%209%20LB/Ch9b_print.html. Acessado em 02/12/2013. vi, 8

[18] T. M. Cover. Geometrical and Statistical Properties of Systems of Linear Inequalities with Applications in Pattern Recognition. *Electronic Computers, IEEE Transactions on*, EC-14(3):326–334, 1965. 43

[19] M. M. Cox, J. A. Doudna, and M. O'Donnell. *Biologia Molecular. Princípios e Técnicas*. Artmed, 2013. 15

[20] T. C. C. da Silva. *SOM-PORTRAIT: um método para identificar RNA não codificador utilizando Mapas Auto Organizáveis*. Monografia de Graduação. Departamento de Ciência da Computação. Universidade de Brasília, 2009. iii, vi, ix, 1, 2, 10, 11, 12, 13, 14

[21] W. Deng et al. Organization of the Caenorhabditis elegans small non-coding transcriptome: Genomic features, biogenesis, and expression. *Genome Res*, 16(1):20–29, 2006. 22

[22] R. Durbin et al. *The Theory Behind Profile HMMS: Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998. 21, 50

[23] S. R. Eddy. Non-coding RNA genes and the modern RNA world. *Nature Reviews Genetics*, 2:919–929, 2001. 10, 12

[24] M. Falaleeva and S. Stamm. Processing of snoRNAs as a new source of regulatory non-coding RNAs. *BioEssays*, 35(1):46–54, 2013. 1, 8, 10, 14, 15, 16

[25] M. S. S. Felipe et al. Transcriptional Profiles of the Human Pathogenic Fungus *Paracoccidioides brasiliensis* in Mycelium and Yeast Cells. *Journal of Biological Chemistry*, 280:24706–24714, 2005. 55

[26] P. Gardner. Metazoan u3 secondary structure. http://en.wikipedia.org/wiki/File:Metazoan_U3_secondary_structure.png. Acessado em 02/12/2013. vi, 11

[27] S. Griffiths-Jones et al. mirbase: tools for microRNA genomics. *Nucleic Acids Research*, 36(Database-Issue):154–158, 2008. 13, 17

[28] A. R. Gruber et al. RNAz 2.0: Improved Noncoding RNA Detection. In *Pacific Symposium on Biocomputing*, pages 69–79, 2010. 49, 54

[29] S. Haykin. *Neural Networks: A Comprehensive Foundation*. Prentice Hall, 1999. vii, ix, 2, 28, 29, 32, 34, 35, 36, 37, 38, 40, 41, 42, 43, 44, 45, 46

[30] J. Hertel, I. L. Hofacker, and P. F. Stadler. SnoReport: computational identification of snoRNAs with unknown targets. *Bioinformatics*, 24(2):158–164, 2008. iii, vi, ix, 3, 5, 10, 14, 16, 17, 18, 20, 21, 22, 57, 61, 66

[31] I. L. Hofacker. Vienna RNA secondary structure server. *Nucleic Acids Research*, 31(13):3429–3431, 2003. 14, 17, 18, 49

[32] I. L. Hofacker et al. Fast folding and comparison of RNA secondary structures (the vienna rna package), 1994. 3, 17

[33] C. Hsu, C. Chang, and C. Lin. A practical guide to support vector classification, 2010. 51

[34] A. Hüttenhofer, P. Schattner, and N. Polacek. Non-coding RNAs: hope or hype? *Trends in Genetics*, 21(5):289 – 297, 2005. 12

[35] Z. P. Huang and other. Genome-wide analyses of two families of snoRNA genes from Drosophila melanogaster, demonstrating the extensive utilization of introns for coding of snoRNAs. *RNA*, 11(8):1303–1316, 2005. 3, 17, 22

[36] A. Hüttenhofer, M. Kiefmann, S. Meier-Ewert, J. O'Brien, H. Lehrach, J. P. Bachellerie, and J. Brosius. RNomics: an experimental approach that identifies 201 candidates for novel, small, non-messenger RNAs in mouse. *EMBO J*, 20(11):2943–2953, 2001. 3, 17

[37] V. Jakkula. Tutorial on support vector machine (SVM). school of EECS. Washington state university, 2006. 32, 34

[38] N. K. Kasabov. *Foundations of Neural Networks, Fuzzy Systems, and Knowledge Engineering*. MIT Press, 1996. 24, 25, 26, 28

[39] S. H. Kim et al. Plant U13 orthologues and orphan snoRNAs identified by RNomics of RNA from Arabidopsis nucleoli. *Nucleic Acids Research*, 38(9):3054–3067, 2010. 16

[40] T. Kin et al. fRNAdb: a platform for mining/annotating functional RNA candidates from non-coding RNA sequences. *Nucleic Acids Research*, 35(Database-Issue):145–148, 2007. 13

[41] S. Kishore and S. Stamm. The snoRNA HBII-52 regulates alternative splicing of the serotonin receptor 2C. *Science*, 311(5758):230–232, 2006. 16

[42] R. Klein, Z. Misulovin, and S. Eddy. Noncoding RNA genes identified in AT-rich hyperthermophiles. *Proc. Natl Acad. Sci.*, 56(99):7542–7547, 2002. 14

[43] E. S. Lander et al. Initial sequencing and analysis of the human genome. *Nature*, 409:860–921, 2001. 10

[44] D. Leja. Transfer RNA (tRNA). http://www.genome.gov/dmd/img.cfm?node=Photos/Graphics&id=85250. Acessado em 02/12/2013. vi, 9

[45] L. Lestrade and M. J. Weber. snoRNA-LBME-db, a comprehensive database of human H/ACA and C/D box snoRNAs. *Nucleic Acids Research*, 34(suppl 1):D158–D162, 2006. 15, 17

[46] X. H. Liang et al. A genome-wide analysis of C/D and H/ACA-like small nucleolar RNAs in *Leishmania major* indicates conservation among trypanosomatids in repertoire and in their rRNA targets. *Eukaryot. Cell*, 6:361–377, 2007. 22

[47] A. M. Lima, V. F. Onuchic, and A. M. Durham. Procura de padrões estruturais em RNA utilizando grafos. In *Curso de Verão 2011 - Bioinformática - USP*, pages 315–318. USP, 2011. 11, 13

[48] C. Lin. Binary-class cross validation with different criteria. http://www.csie.ntu.edu.tw/ cjlin/libsvmtools/eval/index.html Acessado em 10/02/14. 51

[49] C. Liu et al. NONCODE: an integrated knowledge database of non-coding RNAs. *Nucleic Acids Research*, 33(Database-Issue):112–115, 2005. 12

[50] T. M. Lowe and S. R. Eddy. A computational screen for methylation guide snoRNAs in yeast. *Science*, 283(5405):1168–1171, 1999. 23

[51] A. Machado-Lima. *Predição de RNAs não codificantes e sua aplicação na busca do componente RNA da telomerase*. Tese de Doutorado em Bioinformática. Universidade de São Paulo, 2006. ix, 1, 11, 12, 13

[52] A. Machado-Lima et al. Computational methods in noncoding RNA research. *Journal of Mathematical Biology*, 56(1-2):15–49, 2008. 13, 14

[53] T. R. Mercer, M. E. Dinger, and J. S. Mattick. Long non-coding RNAs: insights into functions, 2009. ix, 12

[54] T. M. Mitchell. *Machine Learning*. McGraw-Hill ScienceEngineeringMath, 1997. 2, 24

[55] A. W. Moore. Suport Vector Machines. Tutorial Slides. School of Computer Science. Carnegie Mellon University. Disponível em: www.cs.cmu.edu/ awm, year = 2003. vii, 33

[56] A. Mosig, K. Sameith, and P. F. Stadler. Fragrep: An efficient search tool for fragmented patterns in genomic sequences. *Genomics, Proteomics & Bioinformatics*, 4(1):56–60, 2006. 17

[57] E. P. Nawrocki and S. R. Eddy. Infernal 1.1: 100-fold faster RNA homology searches. *Bioinformatics*, 29(22):2933–2935, 2013. 14

[58] Broad Institute of Havard and MIT. *Paracoccidioides brasiliensis* sequencing project. http://www.broadinstitute.org/annotation/genome/paracoccidioides_brasiliensis /MultiDownloads.html Acessado em 17/02/14. 56

[59] K. C. Pang et al. RNAdb 2.0 - an expanded database of mammalian non-coding RNAs. *Nucleic Acids Research*, 35(Database-Issue):178–182, 2007. 13

[60] J. Pevsner. *Bioinformatics and Functional Genomics*. John Wiley & sons, inc. 2, 5

[61] T. Phillips. sirna. biotech. http://biotech.about.com/od/glossary/g/siRNA.htm Acessado em 14/02/2014. ix, 12

[62] P. Refaeilzadeh, L. Tang, and H. Liu. Cross-Validation. In Ling Liu and M. Tamer Özsu, editors, *Encyclopedia of Database Systems*, pages 532–538. Springer US, 2009. 47, 48

[63] E. Rivas and S. R. Eddy. Secondary structure alone is generally not statistically significant for the detection of noncoding RNAs. *Bioinformatics*, 16(7):583–605, 2000. 14

[64] P. Schattner, S. Barberan-Soler, and T. M. Lowe. A computational screen for mammalian pseudouridylation guide H/ACA RNAs. *RNA*, 12(1):15–25, 2006. 16

[65] P. Schattner et al. Genome-wide searching for pseudouridylation guide snoRNAs: analysis of the Saccharomyces cerevisiae genome. *Nucleic Acids Research*, 32(14):4281–4296, 2004. 23

[66] M. Schmitz. Shuffleseq. emboss. http://emboss.sourceforge.net/apps/release/6.0/emboss/apps/shuffleseq.html Acessado em 16/02/14. 55

[67] M. S. Scott et al. Human box C/D snoRNA processing conservation across multiple cell types. *Nucleic Acids Reseach*, 40(8):3676–3688, 2012. 15, 16, 61

[68] J. C. Setubal and J. Meidanis. *Introduction to Computational Molecular Biology*. PWS Publishing Company, Boston, MA, 2000. 1, 5, 6, 10

[69] M. Szymanski, J. Barciszewski, and V. A. Erdman. *Noncoding RNAs: Molecular Biology and Molecular Medicine, chapter Riboregulators*. Springer, 2003. 1, 10, 12

[70] J. C. Venter et al. The sequence of the human genome. *Science*, 291(5507):1304–1351, 2001. 10

[71] P. Vitali, E. Basyuk, E. Le Meur, E. Bertrand, F. Muscatelli, J. Cavaillé, and A. Huttenhofer. ADAR2-mediated editing of RNA substrates in the nucleolus is inhibited by C/D small nucleolar RNAs. *J Cell Biol*, 169(5):745–753, 2005. 17

[72] S. Washietl et al. Structured RNAs in the ENCODE selected regions of the human genome. *Genome Res*, 17(6):852–864, 2007. 14

[73] J. D. Watson and F. H. C. Crick. Molecular struture of nucleic acids. *Nature*, 171(4356):737–738, 1953. 1

[74] Wikimedia. Difference DNA RNA. http://commons.wikimedia.org/wiki/File:Difference_DNA_RNA-EN.svg Acessado em 02/12/2013. vi, 7

[75] J. Xie et al. Sno/scaRNAbase: a curated database for small nucleolar RNAs and Cajal body-specific RNAs. *Nucleic Acids Research*, 35(Database-Issue):183–187, 2007. 13

[76] J. Yang et al. snoSeeker: an advanced computational package for screening of guide and orphan snoRNA genes in the human genome. *Nucleic Acids Research*, 34(18):5112–5123, 2006. vi, 15, 16, 22, 66

[77] A. Zemann et al. Evolution of small nucleolar RNAs in nematodes. *Nucleic Acids Research*, 34(9):2676–2685, 2006. 22

[78] M. Zuker and P. Stiegler. Optimal computer folding of large RNA sequences using thermodynamics and auxiliary information. *Nucleic acids research*, 9, 1981. 13